SHARP®

PORTABLE COMPUTER

MODEL

PC-2500

OPERATION MANUAL

WARNING: THIS EQUIPMENT HAS BEEN CERTIFIED TO COMPLY WITH THE LIMITS FOR A CLASS B COMPUTING DEVICE, PURSUANT TO SUBPART J OF PART 15 OF FCC RULES. ONLY PERIPHERALS (COMPUTER INPUT/OUTPUT DEVICES, TERMINALS, PRINTERS, ETC.) CERTIFIED TO COMPLY WITH THE CLASS B LIMITS MAY BE ATTACHED TO THIS COMPUTER. OPERATION WITH NON-CERTIFIED PERIPHERALS IS LIKELY TO RESULT IN INTERFERENCE TO RADIO AND TV RECEPTION.

This equipment generates and uses radio frequency energy and if not installed and used properly, that is, in strict accordance with the manufacturer's instructions, may cause interference to radio and television reception. It has been type tested and found to comply with the limits for a Class B computing device in accordance with the specifications in Subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference in a residential installation. However, there is no guarantee that interference will not occur in a particular installation. If this equipment does cause interference to radio or television reception which can be determined by turning the equipment off and on, the user is encouraged to try to correct the interference by one or more of the following measures:

Reorient the receiving antenna.
Relocate the computer with respect to the receiver.
Move the computer away from the receiver.
Plug the computer into a different outlet so that computer and receiver are on different branch circuits.

If necessary, the user should consult the dealer or an experienced radio/television technician for additional suggestions. The user may find the following booklet prepared by the Federal Communications Commission helpful:
"How to Identify and Resolve Radio-TV Interference Problems".
This booklet is available from the U.S. Government Printing Office, Washington, D.C., 20402,
Stock No. 004-000-00345-4
BILLING CODE 6712-01-M

A shielded I/F cable is required to insure compliance with FCC regulation for Class B computing equipments.

FOR YOUR RECORDS....

For your assistance in reporting this electronic calculator in case of loss or theft, please record below the model number and serial number which are located on the bottom of the unit.
Please retain this information.

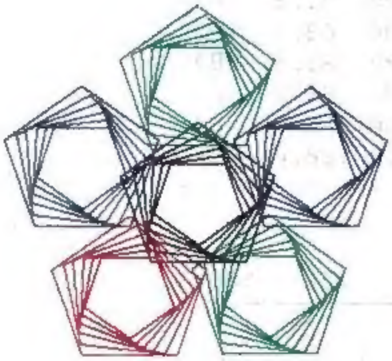
Model Number PC2500 Serial Number 61007392

Date of Purchase ______ Place of Purchase ______

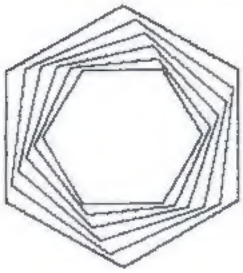
COLOR PRINTOUTS

★ COMPUTER GRAPHIC DESIGNS
(Refer to page 255)

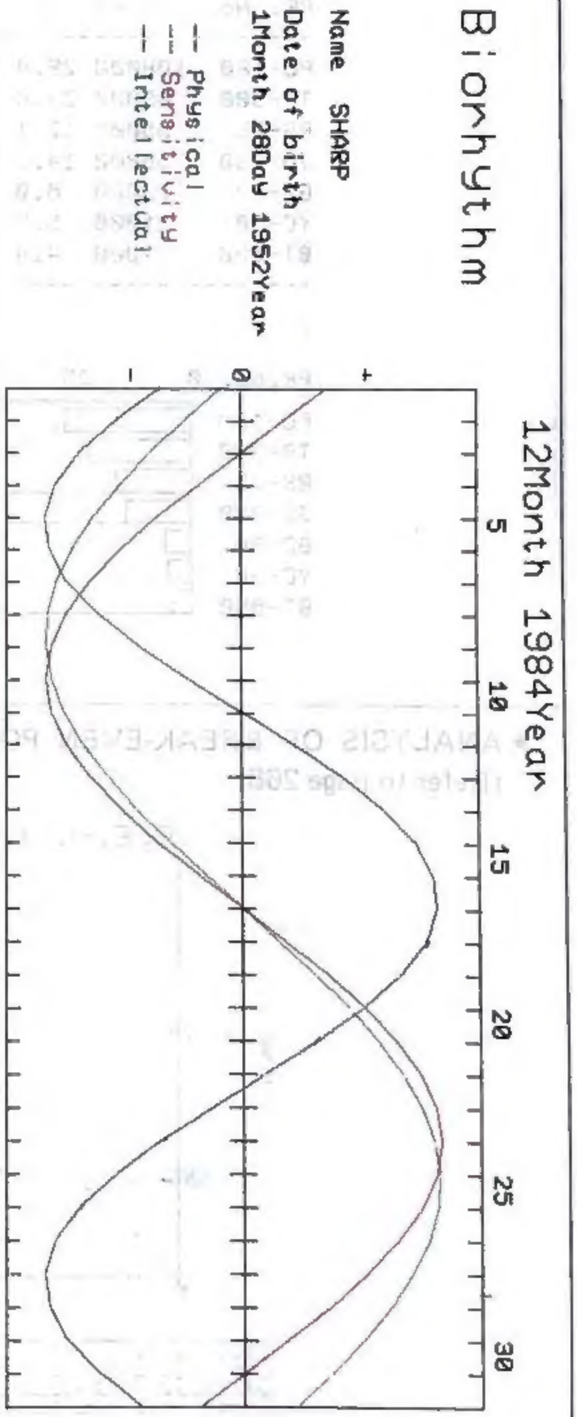
Example 1



Example 2



★ BIORHYTHM
(Refer to page 259)



Biorhythm

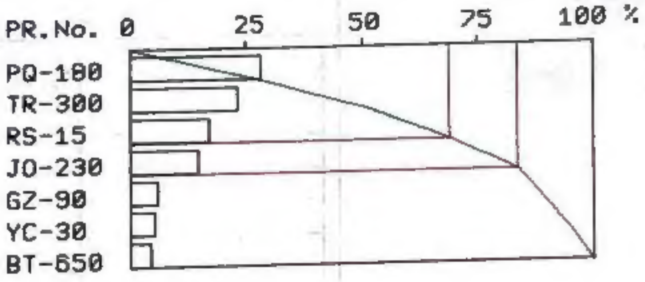
Name SHARP
Date of birth
1Month 28Day 1952Year

12Month 1984Year

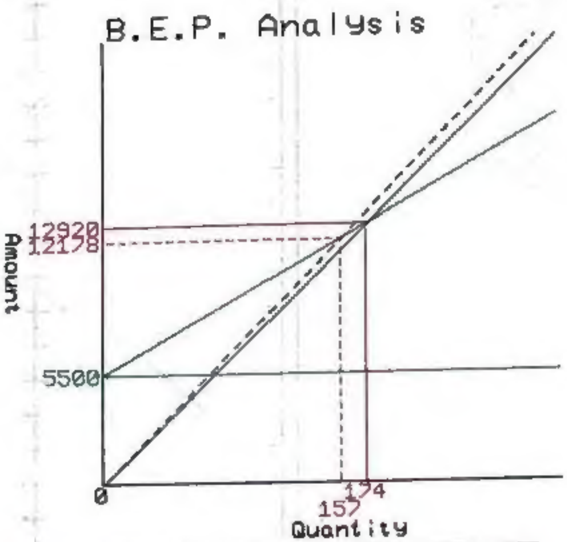
COLOR PRINTOUTS

★ ABC ANALYSIS (Refer to page 263)

```
           ** A B C Analysis **

PR. No.  Sales   %      Sum     %    Rank
-------------------------------------------
PQ-180  108000 28.4   108000   28.4  (A)
TR-300   89000 23.4   197000   51.9  (A)
RS-15    65000 17.1   262000   69.1  (A)
JO-230   56000 14.7   318000   83.9  (B)
GZ-90    23000  6.0   341000   89.9  (C)
YC-30    21000  5.5   362000   95.5  (C)
BT-650   17000  4.4   379000  100.0  (C)
-------------------------------------------
```



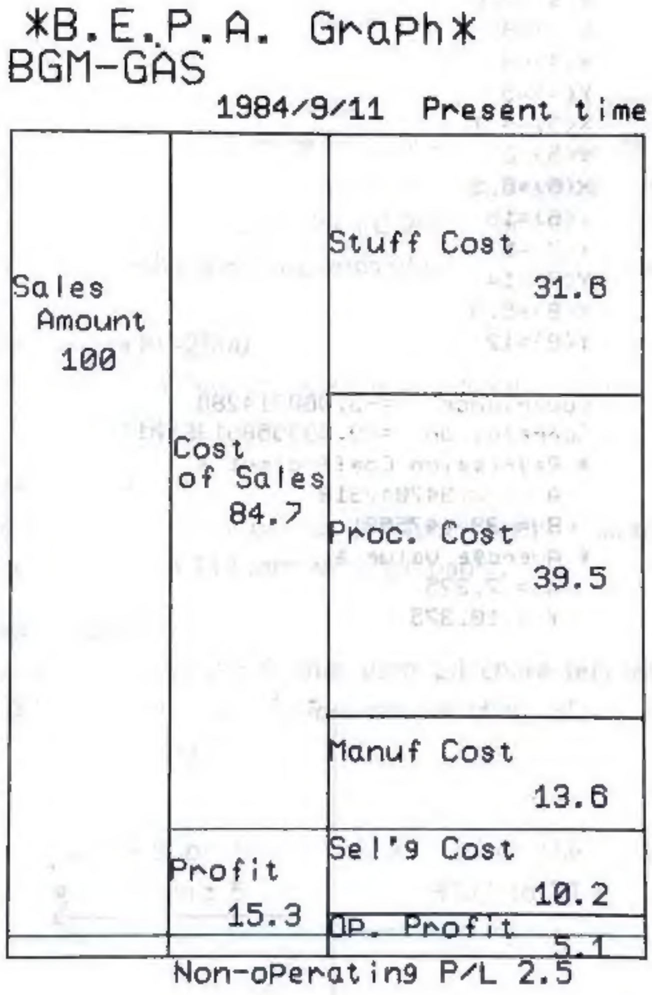
★ ANALYSIS OF BREAK-EVEN POINT OF PROFIT/LOSS (Refer to page 268)



Sales U.P.	74	72.5
Variable U.P.	42.5	42.5
Sales Quantity	200	200
Fixed Cost	5500	5500
Sales Amount	14800	15500
Variable Cost	8500	8500
B.E.P. Amount	12920	12128
B.E.P. Quantity	174	157
Profit	800	1500

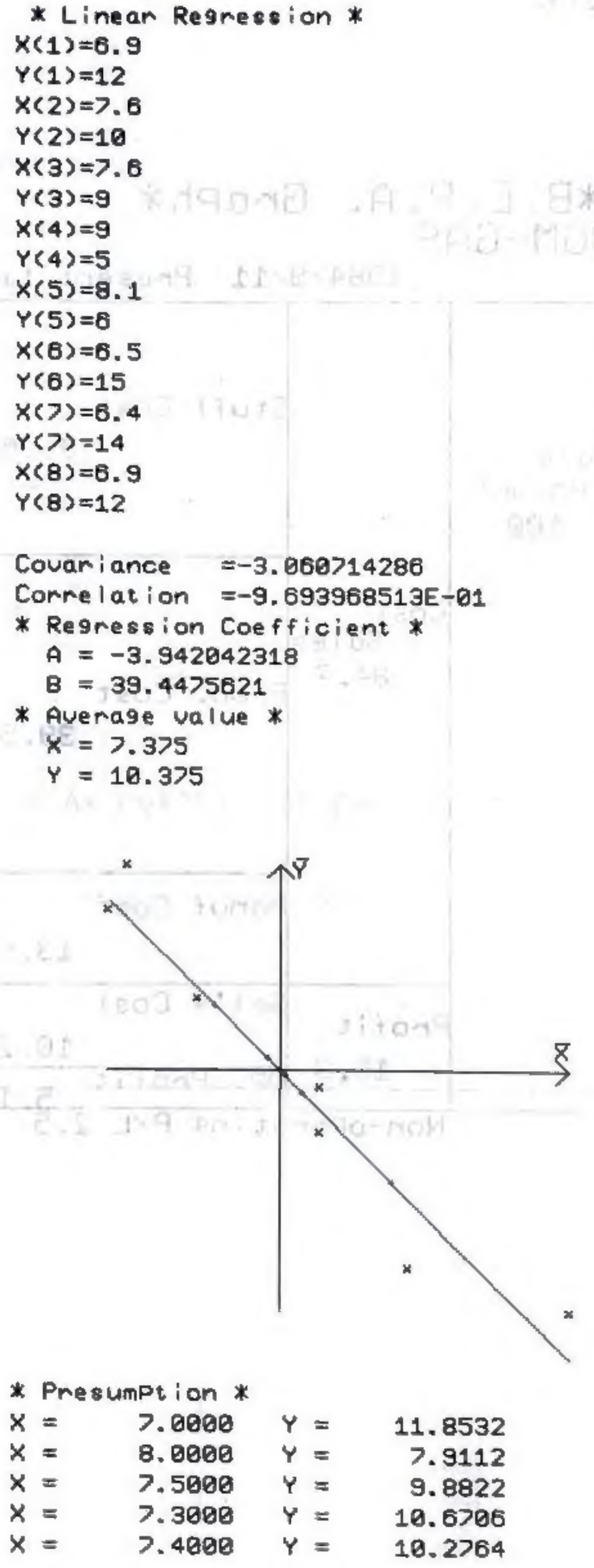
COLOR PRINTOUTS

★ ANALYSIS GRAPH FOR PROFIT AND LOSS CALCULATION (Refer to page 273)



COLOR PRINTOUTS

★ LINEAR REGRESSION AND PLOT (Refer to page 289)



```
 * Linear Regression *
X(1)=6.9
Y(1)=12
X(2)=7.6
Y(2)=10
X(3)=7.6
Y(3)=9
X(4)=9
Y(4)=5
X(5)=8.1
Y(5)=6
X(6)=6.5
Y(6)=15
X(7)=6.4
Y(7)=14
X(8)=6.9
Y(8)=12

Covariance    =-3.060714286
Correlation  =-9.693968513E-01
* Regression Coefficient *
  A = -3.942042318
  B = 39.4475621
* Average value *
  X = 7.375
  Y = 10.375
```

```
* Presumption *
X =      7.0000    Y =     11.8532
X =      8.0000    Y =      7.9112
X =      7.5000    Y =      9.8822
X =      7.3000    Y =     10.6706
X =      7.4000    Y =     10.2764
```

Welcome to the World of Portable Computing!

- Few industries in the world today can match the rapid technological growth recorded in personal computing. Only a short time ago computers filled huge rooms, required Ph.D.'s to program them, and cost thousands of dollars. Now they fit in the palm of the hand, are easy to program, and cost so little that nearly everyone can afford them.

SHARP's PC-2500, the newest and most advanced portable computer, brings you the latest state-of-the-art computing features in a book-sized unit.

The PC-2500 is your first step into the exciting new world of portable computing. It requires no prior computer experience because of its easy-to-use, built-in business software. Ease of operation is ensured by a full-sized keyboard, a built-in printer, and a large LCD panel. In this way, the PC-2500 combines the convenience of pocket computers with the advanced functions of desk-top computers.

- **Business software for the PC-2500**
  The PC-2500's built-in business software is a valuable aid for those unfamiliar with computer programming.

- **Four-color plotter printer**
  The built-in, four-color, plotter printer produces easily readable text and eye-pleasing tables and graphs on 114 mm wide roll paper.

- **Large liquid crystal display**
  The large capacity LCD displays 4 lines each 24 characters long or graphs 150 dots wide and 32 dots high. All inputs can be checked for accuracy prior to program execution or printing.

- **RAM cards for storage**
  Inserting an optional 8 KB or 16 KB RAM card into the expansion slot of the PC-2500 expands the standard 5 KB built-in RAM to 21 KB. The RAM cards have built-in batteries to retain programs and data even when detached from the PC-2500.

- **Easy-to-use keyboard**
  The PC-2500 typewriter-style keyboard and independent 10-key numeric pad allow quick, precise, and convenient data input.

- **Function keys**
  When using the PC-2500's business software, the function keys allow you to interrupt input to do a different job.

- **Interfaces for peripheral devices**
  The PC-2500 is equipped with a cassette tape recorder interface for external storage and a serial interface for data communication.

TABLE OF CONTENTS

CHAPTER 1
HOW TO USE THIS MANUAL

This manual is designed to introduce you to the capabilities and features of your PC-2500 and serve as a valuable reference tool. Whether you are a "first-time user" or an "old hand" with computers, you should acquaint yourself with the PC-2500 by reading and working through chapters 2 and 3.

The first-time user with no knowledge of BASIC programming can nevertheless take advantage of the outstanding features of this new SHARP computer to quickly make such things as GRAPHS, TABLES and TELEPHONE LISTS without knowing BASIC programming.

If you are familiar with BASIC programming and wish to incorporate the computer into your business operations right away, skip to the BUSINESS SOFTWARE section at the end of the manual.

If you wish to learn BASIC or make your own programs, read chapters 4 through 6.

* Chapter 2 describes the physical features of the PC-2500.
* Chapter 3 demonstrates its use as a scientific calculator.
* Chapter 4 defines some terms and concepts essential for BASIC programming and explains how these concepts apply specifically to the PC-2500.
* Chapter 5 introduces you to the PC-2500's BASIC programming, showing you how to enter, correct, and run programs.
* Chapter 6 describes some shortcuts that make your new computer easier and more enjoyable to use.

Experienced BASIC programmers should read Chapter 9 to learn how BASIC is implemented on the PC-2500. Since every BASIC dialect is somewhat different, read through this material at least once before programming.

Chapter 9 is a reference section listing all the verbs, commands, and functions of BASIC in convenient alphabetical order.

If you have never programmed BASIC before, we suggest that you first buy a separate book on BASIC programming or attend a BASIC class. This manual is not intended to teach you programming.

The remainder of the manual consists of:

* Chapter 7 — Basic information on the built-in Printer and Cassette Interface.
* Chapter 8 — Use of the optional CE-201M/202M RAM Card.
* Chapter 10 — A troubleshooting guide to help you solve some operating and programming problems.
* Chapter 11 — Care and maintenance of your new computer.

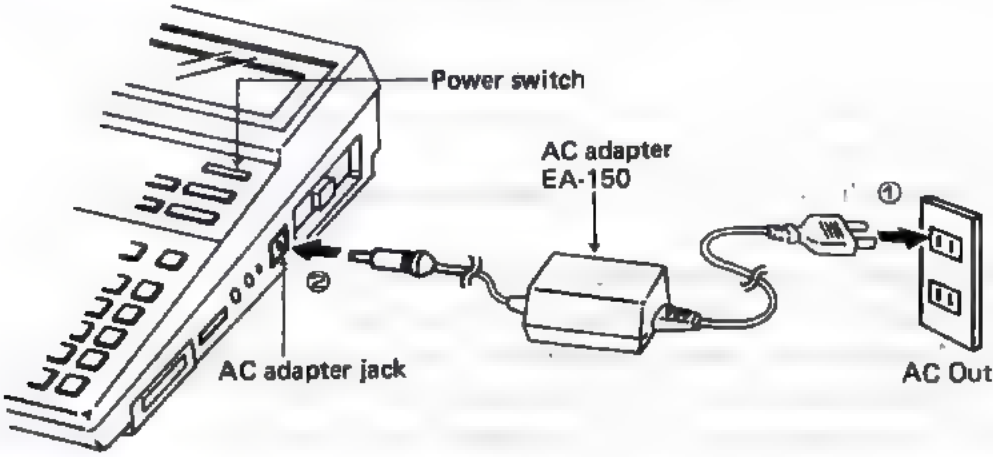
Detailed appendices at the end of the manual provide you with useful charts and describe special uses of the PC-2500.

POWER SUPPLY

The PC-2500 can be operated by using the built-in rechargeable batteries as its power supply. The rechargeable batteries can be charged with the supplied AC adapter (EA-150) by plugging the adapter into an AC outlet.

How to Charge

Turn off the power switch on the PC-2500 and connect the AC adapter (EA-150) by following arrows (1) and then (2) shown below.



With the AC adapter connected and the PC-2500 turned off, the batteries will be completely charged in about 15 hours. Fully charged batteries are capable of printing approximately 450 lines (where twenty 5s are printed in each line using character size "b").
Or, the graph shown on page 304 can be drawn approximately 11 times.
As long as the printer is not used, the batteries can be fully charged in about 15 hours even while the PC-2500 is used.

When to Charge

The low-battery lamp lights when the voltage of the built-in rechargeable batteries becomes low. Turn off the power switch and immediately charge the batteries.

If the PC-2500 is not charged, the stored programs or data may be cleared.

- If the PC-2500 is charged with the power on, the low-battery lamp will still be on after charging is completed. First turn it off and then on again to clear the low-battery state, and then use.
- If the low-battery lamp lights when the power switch is turned on, the pen holder in the printer may rotate when the switch is turned off immediately.

Note: If the low-battery lamp lights during printing, the pen may stay in contact with the paper (roll paper). Turn off the power switch, and then on again, and then charge.

The following also indicate low battery voltage:

① When the display, seen from the front, becomes faint and hard to read even though the display adjustment knob (Contrast control on the right side of the PC-2500) is turned to its darkest setting.

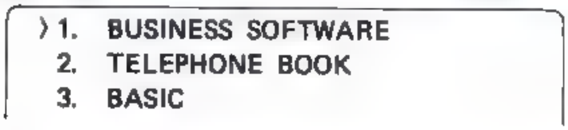
② When ERROR 8 is displayed while the built-in printer is used in the BASIC mode. (However, ERROR 8 is also displayed when an abnormal condition occurs at the printer, tape recorder I/O, or serial I/O.)

③ When "low battery" is displayed while tables are printed or graphs are created by a business software program.

After Charging

After charging, turn on the power switch and check that the following is displayed.

(Menú screen to select Business Software or BASIC)



```
〉1.  BUSINESS SOFTWARE
 2.  TELEPHONE BOOK
 3.  BASIC
```

If the above is not displayed, see page 10 and press the reset switch on the bottom of the PC-2500.

Notes on Charging

- Charge the built-in rechargeable batteries immediately after purchasing the PC-2500 or after long periods of leaving it unused, because the voltage of the batteries will be low due to self-discharge.
- When charging the batteries for the first time or after a long period of storage, the stated operating time may not be obtained even after a full charge. The operating time will return to normal after several charge and discharge cycles.
- Even if not used for a while, the rechargeable batteries can be used for long periods if they are charged at least once or twice every 3 months.
- Avoid charging for 24 hours or more since the performance of the batteries may deteriorate.
- Charge only when the ambient temperature is within 0°C to 40°C. Charging at other temperatures may deteriorate the performance of the batteries.
- The built-in rechargeable batteries have reached the end of their service life when the operating time is reduced by half even after several proper charge and discharge cycles.

Notes on Using the AC Adapter

- Use only AC adapter, model EA-150. The EA-150 can only be used to charge the rechargeable batteries built into the PC-2500. The batteries may burst if a charger other than the EA-150 is used or if the EA-150 is used to charge other types of batteries.

If an Abnormal Condition Occurs

An abnormal condition may occur, where all the keys and switches (including the [illegible] key and power switch) become inoperative, when the PC-2500 is subjected to strong external noise or shock during use.

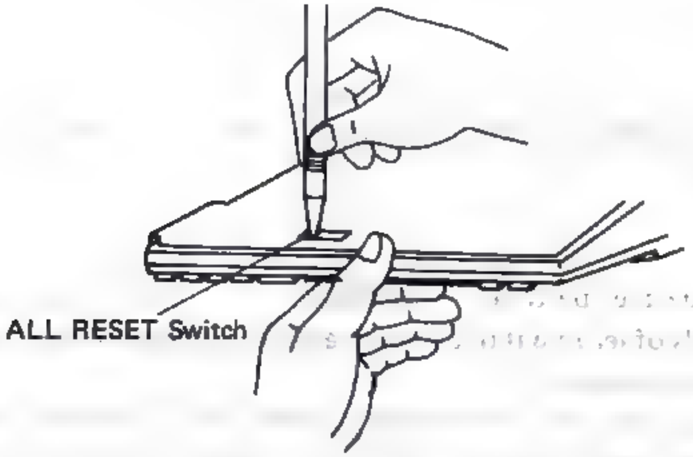
If this occurs, turn on the power switch and perform either of the following.

① **To preserve the stored program and data**

Press the reset switch while holding down the space bar (long key at the bottom of the keyboard with nothing written on it).

If the abnormal condition occurs again, perform (2) and enter the program again.

Note: The program and data are not retained if the reset switch is pressed while a key other than the space bar is held down or while the space bar is held down with another key.

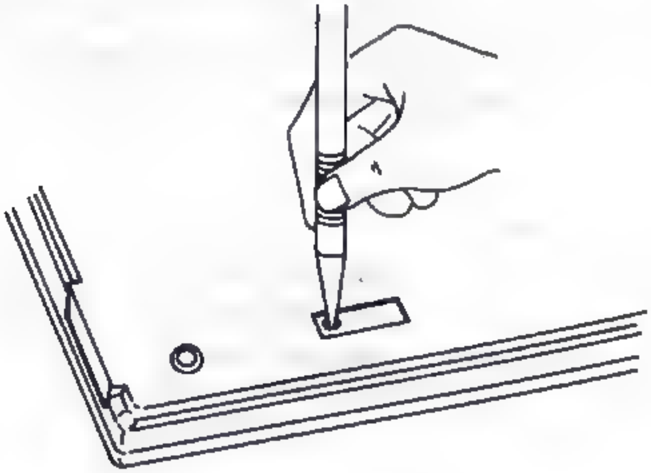


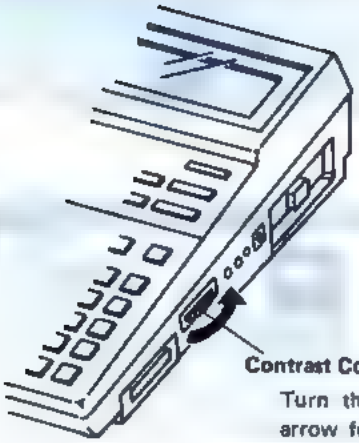
Press the reset switch with a pointed object such as a ball-point pen. Do not use a mechanical pencil (since the lead may break) or a sharp tipped object such as a needle.

② **To completely clear the program and data**

Press the reset switch only. The program and data are completely cleared.

Note: If a RAM card is mounted, the programs and data in the RAM card and the PC-2500 will be cleared. The programs and data can be retained by pressing the reset switch as described in (1).



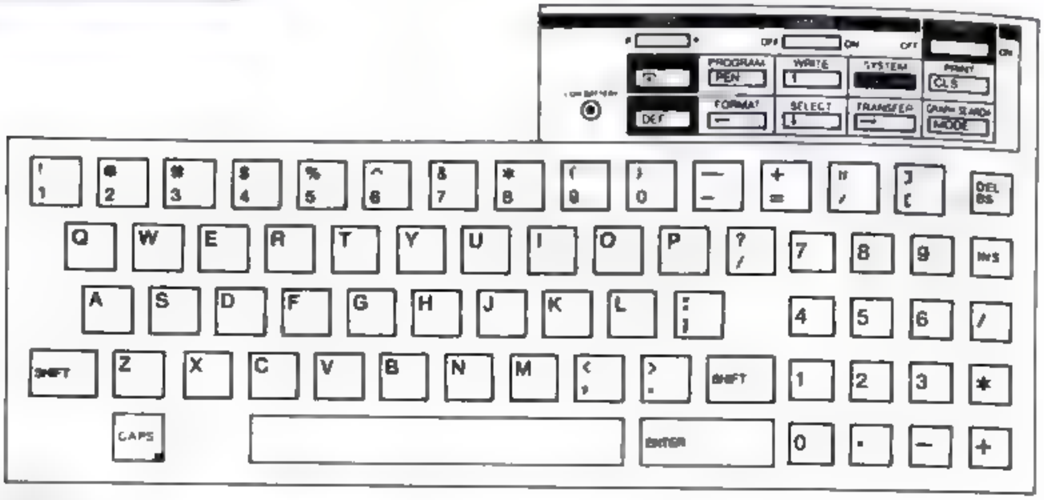


Contrast Control

Turn the control in the direction of the arrow for a darker display, and turn it in the opposite direction for lighter display. Adjust so that the display is easy to see.

CHAPTER 2 INTRODUCTION TO THE PC-2500

Description of Keys



POWER
OFF ON **Power Switch:**

This switch is used to turn on and off the PC-2500.
The following is displayed after the power switch is turned on:
(Menu screen to select Business Software or BASIC)

```
〉1. BUSINESS SOFTWARE
 2. TELEPHONE BOOK
 3. BASIC
```

This display has the following meaning:

- Press the 1 key to run the Business Software.
- Press the 2 key to recall the "telephone book."
- Press the 3 key to use the BASIC language. (Changes to the BASIC mode.)

If the ENTER key is pressed instead of the 1, 2, or 3 key, the function indicated by the 〉 mark is executed.
The 〉 mark can be moved up or down by using the ↑ key or the ↓ key, respectively.

A ~ Z Alphabet Keys:

Press these keys to enter letters. These keys are arranged in a typewriter layout.
Lower case letters are normally entered when these keys are pressed. Upper case letters are entered by pressing these keys while holding down the SHIFT key.
The entry method of upper case letters and lower case letters can be

reversed by pressing the CAPS key.
Therefore, once you press the CAPS key and the green lamp on the CAPS key lights (CAPS symbol is shown on the display unit), upper case letters are entered if you press only the alphabet keys and lower case letters are entered if you press the alphabet keys while holding down the SHIFT key.

□ **Space Bar:**
Press this bar (key) to enter a space.

1 2 ~ 0 **Number/Symbol Keys:**
Press these keys to enter numbers. (Numbers can also be entered with the numeric keys.) Press these keys while holding down the SHIFT key to enter the symbols written at the top of the keys.

Symbol Keys: (=, ±, /,], ?, :, <, >)
Press these keys to enter the symbols written on the lower half of the keys. Press these keys while holding down the SHIFT key to enter the symbols written at the top of the keys.

SHIFT **SHIFT Key:**
Press an alphabet key while holding down the SHIFT key to enter upper case letters. (Lower case characters if in the CAPS mode.) Press a number/symbol or symbol key while holding down the SHIFT key to enter the symbol written at the top of the key. Similarly, press the DEL INS key while holding down the SHIFT key to execute the function (delete) written at the top of the key.

CAPS **Capital (CAPS) Key:**
If the CAPS key is pressed once, the green lamp (CAPS lamp) on the CAPS key lights. At the same time, the CAPS symbol appears on the display unit. If the CAPS key is pressed again, the CAPS lamp goes off and the CAPS symbol clears.
Usually, lower case letters are entered if the alphabet keys are pressed and upper case letters are entered if the alphabet keys are pressed while holding down the SHIFT key.
Press the CAPS key and light the CAPS lamp to reverse the entry method of lower case and upper case letters.

ENTER **Enter Key:**
Press this key to specify the end of a program line and write the line to memory.
This key is also used to execute programs and manual calculations.
In the Business Software, this key is used to enter data or advance to the next step.

0 ~ 9 **Numeric Keys:**
These keys are used to enter numbers and numeric values.

[•] **Decimal Point Key:**
Press this key to enter the decimal point.

[−] **Minus Key:**
Press this key to enter the subtraction operator or minus sign.

[+] **Plus Key:**
Press this key to enter the addition operator or plus sign.

[✱] **Multiplication Key:**
Press this key to enter the multiplication operator.

[/] **Division Key:**
Press this key to enter the division operator.

[INS] **Insert (INS) Key:**
Press this key to insert a space (displayed as ⌶) at the position indicated by the cursor. You can INSert a new character into this space.

[DEL BS] **Backspace/Delete Key:**
Press this key to delete the character to the left of the position indicated by the cursor.
Press this key while holding down the [SHIFT] key to delete the character at the position indicated by the cursor.

REMOTE
OFF [□□] ON **Remote Switch:**
This switch turns the remote control function on or off and is used to start and stop the tape recorder when recording programs on tape or loading programs from tape.

PRINT
P [□□] • **Print Switch:**
Set this switch to the "P" position to print equations and results of manual calculations.
If a print-out of equations and results of manual calculations is not needed, set this switch to the " • " position.

[↟] **Paper Feed Key:**
Press this key to advance the paper. The paper continues to advance while this key is held down.

[DEF] **Definable Key:**
This key is used to start programs, recall reserved contents, or enter various modes in the Business Software.

[CLS] **Key:**
This is used to clear entries or the display.
It is also used to clear errors.
Pressing this key while holding down the [SHIFT] key clears various states in the PC-2500.
See page 305 for its function during execution of the Business Software.

SYSTEM
[ON/BRK] **Key:**
This key is used to turn on the power after the power has been shut off by the auto-power off function. This key is also used to temporarily stop program execution. During execution of the CLOAD or SAVE command, this key stops execution.
When this key is pressed while holding down the SHIFT key, the menu screen to select the Business Software or BASIC language will appear on the display unit.
See page 305 for its function during execution of the Business Software.

[illegible]
[MODE] **Key:**
This key is used to switch the modes (RUN, PROgram, or RESERVE) while in BASIC. The mode switches alternately between the RUN mode and the PROgram mode each time the [MODE] key is pressed. If the [MODE] key is pressed while holding down the [SHIFT] key, the mode switches alternately between the RUN mode and the RESERVE mode.
See page 306 for its function during execution of the Business Software.

TRANSFER
[→] **Key:**
This key moves the cursor to the right without deleting previously entered characters.
See page 306 for its function during execution of the Business Software.

FORMAT
[←] **Key:**
This key moves the cursor to the left without deleting previously entered characters.
See page 305 for its function during execution of the Business Software.

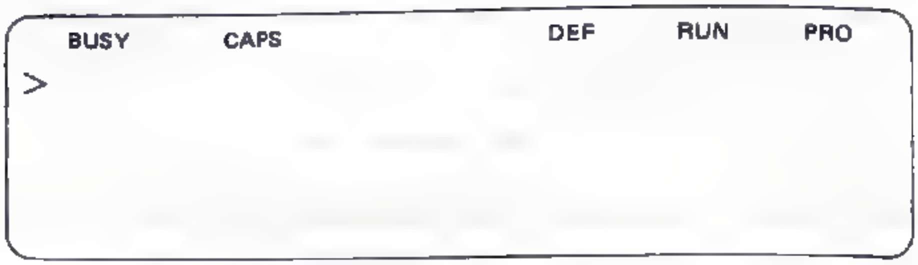
WRITE
[↑] **Key:**
Press this key to display the previous line.
See page 306 for its function during execution of the Business Software.

SELECT
[↓] **Key:**
Press this key to display the next line. See page 306 for its function during execution of the Business Software.

PROGRAM
[PEN] **Key:**
Press this key while holding down the [SHIFT] key to enter the pen change mode. The pen holder moves to the right end.
Now, press the [PEN] key to rotate the pen holder so that the pen can be changed.
Again press the [PEN] key while holding down the [SHIFT] key to clear the pen change mode. The pen moves to the left end.

- See page 305 for its function during execution of the Business Software.

Description of Display



The PC-2500 has a 150 x 32 programmable, dot-matrix liquid crystal display. This display shows up to 24 characters per a line and total 4 lines. Each character occupies a 5 x 7 dot matrix.

For graphic purposes, the entire display may be utilized as a 150 x 32 dot matrix. Individual dots within any of 150 columns may be energized to create graphics, figures, or special symbols.

The display consists of:

> The cursor. This symbol (the underline) tells you the location of the next character to be typed in. As you begin typing, the cursor replaces the prompt. The cursor is also used to position the computer over certain characters when using the INSert and DELete functions.

RUN RUN indicator. This indicator tells you the operational mode of the PC-2500 is in the RUN mode.

PRO PROgram indicator. This indicator tells you the operational mode of the PC-2500 is in the programming mode.

Note: If neither RUN nor PRO indicator can be found on the display, the PC-2500 is in the reserve (RSV) mode.

DEF Definable Mode Indicator. This symbol light up when you press the DEF key.

CAPS This symbol can be displayed or cleared by pressing the CAPS key.
The entry method of upper case and lower case letters when this symbol is displayed is opposite from when it is not displayed.

BUSY Symbol indicating that a program is being executed in the BASIC mode.

For Displays Exceeding 4 Lines

The display unit of the PC-2500 consists of 4 lines (24 characters per line). Key inputs or calculated results are displayed from the top line of the display. If the characters to be displayed exceed 4 lines, the displayed contents will be moved up by 1 line (the first displayed line will move off the top of the screen and disappear).

CHAPTER 3
USING THE PC-2500 AS A CALCULATOR

Now that you are familiar with the layout and components of the SHARP PC-2500, we will begin investigating the exciting capabilities of your new computer.

Because the PC-2500 allows you the full range of calculating functions, plus the increased power of BASIC programming abilities (useful in more complex calculations), it is commonly referred to as a "smart" calculator. That, of course, makes you a "smart" user!

(Charge the built-in rechargeable batteries before using the PC-2500.)

Start up

Turn off the power switch and then on. The following will be displayed:

```
> 1. BUSINESS SOFTWARE
  2. TELEPHONE BOOK
  3. BASIC
```

Calculations are usually performed in the BASIC mode. Press the [3] key to enter the BASIC mode.

The following will be displayed after pressing the [3] key:

```
RUN MODE
>
```

This indicates the RUN Mode (where calculation and programs can be executed). The ">" mark is called a prompt and indicates that the PC-2500 is ready for operation.

Auto Off

In order to conserve on battery wear, the PC-2500 automatically powers down when no keys have been pressed for about 11 minutes. (Note: The PC-2500 will not AUTO OFF while you are executing a program.)

To restart the computer after an AUTO OFF, press the [BRK] key. All settings will be exactly as they were when the AUTO OFF occured.

Some Helpful Hints

Until you get used to your new machine, you are bound to make mistakes while entering data. Later we will discuss some simple ways to correct these mistakes. For now, if you get an Error Message, press the red clear ([CLS]) key and retype the entry. If the computer "hangs up"—you cannot get it to respond at all—press the ALL RESET button (See Chapter 2).

The PROMPT (>) tells you that the PC-2500 is awaiting input. As you enter data the prompt disappears and the CURSOR (_) moves to the right indicating the next available location in the display.

The right [→] and left [←] arrows move the cursor.

Pressing [ENTER] informs the PC-2500 that you are finished entering data and signals the computer to perform the indicated operations. **YOU MUST PRESS [ENTER] AT THE END OF EACH LINE OF INPUT OR YOUR CALCULATIONS WILL NOT BE ACTED UPON BY THE COMPUTER.**

When performing numeric calculations input appears on the left of the display: the results appear on the right of the display.

When you want to enter a capital letter or a symbol on the top half of the key, press the desired key simultaneously with the [SHIFT] key.

Do not use dollar signs or commas when entering calculations into the PC-2500. These characters have special meaning in the BASIC programming language.

In this manual we use Ø to indicate zero, so that you can distinguish between the number (Ø) and the letter (O).

To help get you started entering data correctly, we will show each keystroke necessary to type in the example calculations. When [SHIFT] is used we will represent the desired character in the following keystroke. For example pressing [SHIFT] and [!/1] will produce the ! character. These keystrokes are written [SHIFT] [!].

Be sure to enter CLear after each calculation (unless you are performing serial calculations). CLear erases the display and resets the error condition. It does not erase anything stored in the computer's memory.

Simple Calculations

The PC-2500 performs calculations with ten-digit precision. Turn ON your computer and select BASIC. The PC-2500 will be in the RUN mode. Now try these simple arithmetic examples.

Input | Display

[5] [Ø] [+] [5] [Ø] [ENTER]

```
5Ø+5Ø
                         1ØØ.
```

[1] [Ø] [Ø] [−] [5] [Ø] [ENTER]

```
5Ø+5Ø
                         1ØØ.
1ØØ−5Ø
                          5Ø.
```

[6] [Ø] [*] [1] [Ø] [ENTER]

```
1ØØ-5Ø
                        5Ø.
6Ø*1Ø
                       6ØØ.
```

The display unit of the PC-2500 consists of 4 lines (24 characters per line). Key inputs or calculated results are displayed from the top line of the display. If the characters to be displayed exceed 4 lines, the displayed contents will be moved up by 1 line (the first displayed line will move off the top of the screen and disappear).

Input

Display

[3] [Ø] [Ø] [/] [5] [ENTER]

```
6Ø*1Ø
                       6ØØ.
3ØØ/5
                        6Ø.
```

[1] [Ø] [SHIFT] + [^] [2] [ENTER]

This shows that you must enter [^] while holding down the [SHIFT] key.

```
3ØØ/5
                        6Ø.
1Ø^2
                       1ØØ.
```

[2] [*] [P] [I] [ENTER]

```
1Ø^2
                       1ØØ.
2*PI
               6.2831853Ø7
```

[S] [Q] [R] [6] [4] [ENTER]

```
2*PI
               6.2831853Ø7
SQR 64
                         8.
```

[4] [E] [3] [ENTER]

```
SQR 64
                         8.
4E3
                      4ØØØ.
```

Recalling Entries

Even after the PC-2500 has displayed the results of your calculation, you can edit your last entry. To edit, use the left [←] and right [→] arrows.

The left arrow [←] is used to position the cursor after the last character.

The right arrow [→] is used to position the cursor "on top of" the first character.

Remember that the left and right arrows are also used to position the cursor along a line. The right and left arrows are very helpful in editing (or modifying) entries without having to retype the entire expression.

You will become familiar with the use of the right and left arrows in the following examples. Now, take role of the manager and perform the calculations as we discuss them.

As the head of personnel in a large marketing division, you are responsible for planning the annual sales meeting. You expect 3ØØ people to attend the three day conference. For part of this time, the sales force will meet in small groups. You believe that groups of six would be a good size. How many groups would this be?

Input	Display
[3] [Ø] [Ø] [/] [6] [ENTER]	3ØØ/6 5Ø.

On second thought you decide that groups containing an odd number of participants might be more effective. Recall your last entry using the [←] arrow.

Input	Display
[←]	3ØØ/6_

To calculate the new number of groups you must replace the six with an odd number. Five seems to make more sense than seven. Because you recalled using the [←] arrow, the cursor is positioned at the end of the display. Use the [←] to move the cursor one space to the left.

Input	Display
[←]	3ØØ/█

Notice that after you move the cursor it becomes a flashing block █. Whenever you position the cursor "on top of" an existing character, it will be displayed as the flashing cursor.

Type in a 5 to replace the 6. One caution in replacing characters—once you type a new character over an existing character, the original is gone forever! You cannot recall an expression that has been typed over.

Input	Display
[5]	3ØØ/5_
[ENTER]	6Ø.

Sixty seems like a reasonable number of groups, so you decide that each small group will consist of five participants.

Recall is also useful to verify your last entry, especially when your results do not seem to make sense. For instance, suppose you had performed this calculation:

Input	Display
[CLS] [3] [Ø] [/] [5] [ENTER]	3Ø/5 6.

Even a tired, overworked manager like you realizes that 6 does not seem to be a reasonable result when you are dealing with hundreds of people! Recall your entry using the [→] .

Input	Display
[→]	█Ø/5

Because you recalled using the [→] the flashing cursor is now positioned over the first character in the display. To correct this entry you wish to insert an added zero. Using the [→] , move the cursor until it is positioned over the zero. When making an INSert, you position the flashing cursor over the character before which you wish to make the insertion.

Input	Display
[→]	3█/5

Use the INSert key to make space for the needed character.

Input	Display
[INS]	3 █ Ø/5

Pressing INSert moves all the characters one space to the right, and inserts a bracketed open slot. The flashing cursor is now positioned over this open space, indicating the location of the next typed input. Type in your zero. Once the entry is corrected, display your new result.

Input	Display
[0]	3■Ø/5
[ENTER]	6Ø.

On the other hand, suppose that you had entered this calculation:

Input	Display
[CLS]	
[3] [0] [0] [0] [/] [5] [ENTER]	3ØØØ/5 6ØØ.

The results seem much too large. If you only have 3ØØ people attending the meeting, how could you have 6ØØ "small groups"? Recall your entry using the [→] .

Input	Display
[→]	■ØØØ/5

The flashing cursor is now positioned over the first character in the display. To correct this entry eliminate one of the zeros. Using the [→] move the cursor to the first zero (or any zero). When deleting a character, you position the cursor "on top of" the character to be deleted.

Input	Display
[→]	3■ØØ/5

Now use the DELete key to get rid of one of the zeros.

Input	Display
[SHIFT] + [DEL]	3■Ø/5

Pressing DELete causes all the characters to shift one space to the left. It deletes the character it is "on top of" and the space the character occupies. The flashing cursor stays in the same position indicating the next location for input. Since you have no other changes to make, complete the calculation.

Input	Display
ENTER	3ØØ/5 6Ø.

(Note: Pressing the space key, when the cursor is positioned over a character, replaces the character leaving a blank space. DELete eliminates the character and the space it occupied.)

Errors

Recalling your last entry is essential when you get the dreaded ERROR message. Let us imagine that, unintentionally, you typed this entry into the PC-2500:

Input	Display
CLS 3 Ø Ø / / 5 ENTER	3ØØ//5 ERROR 1

Naturally you are surprised when this message appears! ERROR 1 is simply the computer's way of saying. "I don't know what you want me to do here".
At this time, when the ← key is pressed, the flashing cursor appears at the location where the error occurred.

Input	Display
←	3ØØ/█5

To correct this error use the BS key.

Input	Display
BS ENTER	3ØØ/5 6Ø.

If, upon recalling your entry after an ERROR 1, you find that you have omitted a character, use the INSert sequence to correct it.
When using the PC-2500 as a calculator, the majority of the errors you encounter will be ERROR 1 (an error in syntax). For a complete listing of error messages, see APPENDIX A.

Serial Calculations

The PC-2500 allows you to use the results of one calculation as part of the following calculation.

Part of your responsibility in planning this conference is to draw up a detailed budget for approval. You know that your total budget is $15Ø.ØØ for each attendant. Figure your total budget:

Input	Display
CLS 3 Ø Ø * 1 5 Ø ENTER	3ØØ*15Ø 45ØØØ.

Of this amount you plan to use 15% for the final night's awards presentation. When performing serial calculations it is not necessary to retype your previous results, but DO NOT CLear between entries. What is the awards budget?

Input	Display
* . 1 5	45ØØØ. *. 15_

Notice that as you type in the second calculation (*. 15), the computer automatically displays the result of your first calculation at the left of the screen and includes it in the new calculation. In serial calculations the entry must begin with an operator. As always, you end the entry with ENTER :

NOTE: The % key cannot be used in the calculation. The % key should be used as a character only.
Example: 45ØØØ * 15 SHIFT + % → ERROR 1

Input	Display
ENTER	675Ø.

Continue allocating your budget. The hotel will cater you dinner for $4ØØØ:

Input	Display
– 4 Ø Ø Ø	675Ø. –4ØØØ_
ENTER	275Ø.

Decorations will be $1225:

Input	Display
− 1 2 2 5 ENTER	1525.

Finally, you must allocate $22ØØ for the speaker and entertainment:

Input	Display
− 2 2 Ø Ø ENTER	−675.

Obviously, you will have to change either your plans or your allocation of resources!

Negative Numbers

Since you want the awards dinner to be really special, you decide to stay with the planned agenda and spend the additional money. However, you wonder what percentage of the total budget will be used up by this item. First, change the sign of the remaining sum:

Input	Display
* − 1	−675. * −1_
ENTER	675.

Now you add this result to your original presentation budget:

Input	Display
+ 6 7 5 Ø ENTER	7425.

Dividing by 45ØØØ gives you the percentage of the total budget this new figure represents:

Input	Display
/ 4 5 Ø Ø Ø ENTER	Ø.165

Fine, you decide to allocate 16.5% to the awards presentation.

Compound Calculations and Parentheses

In performing the above calculations, you could have combined several of these operations into one step. For instance, you might have typed both these operations on one line:

675+675Ø/45ØØØ

Compound calculations, however, must be entered very carefully:

675+675Ø/45ØØØ might be interpreted as

$$\frac{675+675Ø}{45ØØØ} \quad \text{or} \quad 675+\frac{675Ø}{45ØØØ}$$

When performing compound calculations, the PC-2500 has specific rules of expression evaluation and operator priority (see APPENDIX D). Be sure you get the calculation you want by using parentheses to clarify your expressions:

(675+675Ø)/45ØØØ   or   675+(675Ø/45ØØØ)

To illustrate the difference that the placement of parentheses can make, try these two examples:

Input	Display
SHIFT + (6 7 5 + 6 7 5 Ø SHIFT +) / 4 5 Ø Ø Ø ENTER	Ø.165
6 7 5 + SHIFT + (6 7 5 Ø / 4 5 Ø Ø Ø SHIFT +) ENTER	675.15

Using Variables in Calculations

The PC-2500 can store up to 26 simple numeric variables under the alphabetic characters A to Z. If you are unfamiliar with the concept of variables, they are more fully explained in Chapter 4. You designate variables with an Assignment Statement:

A = 5
B = –2

You can also assign the value of one variable (right) to another variable (left):

C = A+3
D = E

A variable may be used in place of a number in any calculation.

Now that you have planned your awards dinner, you need to complete arrangements for your conference. You wish to allocate the rest of your budget by percentages also. First you must find out how much money is still available. Assign a variable (R) to be the amount left from the total:

Input	Display
R = 4 5 Ø Ø Ø - 7 4 2 5 ENTER	R=45ØØØ-7425_ 37575.

As you press ENTER the PC-2500 performs the calculation and displays the new value of R. You can display the current value of any variable by entering the alphabetic character it is stored under:

Input	Display
R ENTER	37575.

You can then perform calculations using your variable. The value of (R) will not change until you assign it a new value.

You wish to allocate 6Ø% of the remaining money to room rental:

Input	Display
R * . 6 Ø ENTER	R * .6Ø_ 22545.

Similarly, you want to allocate 25% of your remaining budget to conduct management training seminars:

Input	Display
R * . 2 5 ENTER	9393.75

Variables will retain their assigned values even if the machine is turned OFF or undergoes an AUTO OFF. Variables are lost only when:

- You assign a new value to the same variable.
- You type in CLEAR ENTER (not the clear key).
- You clear the machine using the ALL RESET button.
- The built-in rechargeable batteries are so worn that the variable's contents cannot be maintained.

There are certain limitations on the assignment of variables, and certain programming procedures which cause them to be changed. See Chapter 4 for a discussion of assignment. See Chapter 5 for a discussion of the use of variables in programming.

Chained Calculations

In addition to combining several operators in one calculation, the PC-2500 also allows you to perform several calculations one after the other—without having to press [ENTER] before moving on. You must separate the equations with commas. Only the result of the final calculation is displayed. (Remember too, that the maximum line length accepted by the computer is 8Ø characters including [ENTER].)

You wonder how much money would have been available for rooms if you had kept to your original allocation of 15% for the awards dinner:

Input

[R] [=] [.] [8] [5] [*] [4] [5]
[Ø] [Ø] [Ø] [,] [R] [*]
[.] [6] [Ø]

Display

```
R=.85*45ØØØ, R*.6Ø_
```

Although the computer performs all the calculations in the chain, it displays only the final result:

Input

[ENTER]

Display

```
R=.85*45ØØØ, R*.6Ø
                              2295Ø.
```

To find the value of R used in this calculation, enter R:

Input

[R] [ENTER]

Display

```
                              3825Ø.
```

Scientific Notation

People who need to deal with very large and very small numbers often use a special format called exponential or scientific notation. In scientific notation a number is broken down into two parts.

The first part consists of a regular decimal number between 1 and 1Ø. The second part represents how large or small the number is in powers of 1Ø.

As you know, the first number to the left of the decimal point in a regular decimal number shows the number of 1's, the second shows the number of 1Ø's, the third the number of 1ØØ's, and the fourth the number of 1ØØØ's. These are simply increasing powers of 1Ø:

$1Ø^0=1$, $1Ø^1=1Ø$, $1Ø^2=1ØØ$, $1Ø^3=1ØØØ$, etc.

Scientific notation breaks down a decimal number into two parts: one shows what the numbers are, the second shows how far a number is to the left, or right, of the decimal point. For example:

1234 becomes 1.234 times $1Ø^3$ (3 places to the right)
654321 becomes 6.54321 times $1Ø^5$ (5 places to the right)
.ØØØ125 becomes 1.25 times $1Ø^{-4}$ (4 places to the left)

Scientific notation is useful for many shortcuts. You can see that it would take a lot of writing to show 1.Ø times $1Ø^{87}$ — a 1 and 87 zeros! But, in scientific notation this number looks like this:

$1.Ø \times 1Ø^{87}$ or 1.ØE 87

The PC-2500 uses scientific notation whenever numbers become too large to display using decimal notation. This computer uses the capital letter E to mean "times ten to the":

1234567890ØØØ is displayed as 1.23456789 E 12
.ØØØØØØØØØØØ1 is displayed as 1. E –12

Those of you who are unfamiliar with this type of notation should take some time to put in a few very large and very small numbers to note how they are displayed.

Limits

The largest number which the PC-2500 can handle is ten significant digits, with two digit exponents. In other words the largest number is:

```
9.999999999 E 99= 9999999999ØØØØØØØØØØØØØØØØØØØØØØØØ
                  ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
                  ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
```

and the smallest number is:

```
9.999999999 E –99= .ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
                    ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ
                    ØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØØ9
                    999999999
```

Last Answer Feature

In the case of the serial calculation, you could use the result of the calculation only as the first member of the subsequent calculation formula.
Refer to the following example.

Input	Display
3 [+] 4 [ENTER]	3 + 4 7.
[*] 5	7.*5_
[ENTER]	35.

Press [CLS], then the [↓] or [↑] key. If you operated these keys just after completing the calculation example above, you should see "35." in your display. The numeric data displayed is the result of your last calculation.

The PC-2500 can "remember" the last answer (result) obtained through manual calculation, and recall it on its display with the [↓] or [↑] key.

In the case of the serial calculation described above, you could use the result of the previous calculation only as the first member of the subsequent calculation formula. With the last answer feature, however, you can place the result of the previous calculation in any position of the subsequent calculation.

(Example) Use the result (6.25) of the operation, 5Ø÷8, to compute 12x5÷6.25+24x3÷6.25=:

Input	Display
5Ø [/] 8 [ENTER]	5Ø/8 6.25 ↑ Last answer
12 [*] 5 [/] [↑] (or [↓])	12*5/6.25_ ↑ Last answer recalled
[+] 24 [*] 3 [/] [↑] (or [↓])	12*5/6.25+24*3/6.25_ ↑ Last answer recalled
[ENTER]	21.12

CLS ↓

```
21.12_
```

The last answer is replaced with the result of the previous calculation by performing a manual calculation with the ENTER key.

As shown in this example, the last answer can be recalled anytime and anyplace, but will be replaced with a new last answer resulting from the last calculation.

The last answer is not cleared by the CLS or key operation.

- The last answer cannot be recalled when the computer is not in the RUN mode, program execution is temporarily halted, or the Trace mode is selected.
- The last answer will also be replaced when a program is executed.
- The last answer will be cleared if the power switch is turned off and then on.

Length of Formula

The length of a formula you can put into your computer has a certain limitation. With the PC-2500, up to 79 key strokes can be used to enter a single calculation formula (excluding the ENTER key). If you attempt the 8Øth key stroke, the cursor (█) will start blinking on that character, indicating that the 8Øth key entry is not valid.

Scientific Calculations

The PC-2500 is equipped with the basic functions shown below. Note that the notation of the functions in BASIC may differ from conventional mathematical notations.

Function	Conventional notation	Key operation	Remarks
Trigonometric functions	sin cos tan	SIN COS TAN	
Inverse trigonometric functions	$\sin^{-1}$ $\cos^{-1}$ $\tan^{-1}$	ASN ACS ATN	
Common logarithm	log	LOG	$\log_{10} x$ (logarithm based on 1Ø)
Natural logarithm	ln	LN	$\log_e x$ (logarithm based on e)
Exponential function	e^x	EXP	$e \simeq 2.718281828$

Function	Conventional notation	Key operation	Remarks
Exponentiation		^	A^B for A ^ B
Square root	√‾	SQR	
Degrees (decimal) → degrees (degrees, minutes, seconds) conversion		DMS	Angle conversion (Do not leave out the Ø as in DEG. 5 instead of DEG Ø.5)
Degrees (degrees, minutes, seconds) → degrees (decimal) conversion		DEG	
Integer		INT	In INT (x), obtains the largest integer less than or equal to x.
Absolute	| X |	ABS	In ABS (x), obtains the absolute value of x.
Sign		SGN	Results in 1 when x > Ø, −1 when x<Ø, and Ø when x=Ø for SGN(x).
Pi	π	PI	PI ≃ 3.141592654
Hexadecimal notation →	Decimal notation	&	Converts x to a number in base 1Ø for &x.

Angular unit	Command	Description
Degree	DEGREE	Represents a right angle as 9Ø [°].
Radian	RADIAN	Represents a right angle as π/2 [rad].
Grad	GRAD	Represents a right angle as 1ØØ [g].

These instructions are used to specify angular units in program. For practice, use these instructions to specify angular units in the following calculation examples:

(Example) sin 3Ø° =

(Operation) DEGREE ENTER (Specifies "degree" for angular unit.)

SIN 3Ø ENTER | Ø.5

(Example) $\tan \frac{\pi}{4} =$

(Operation) RADIAN ENTER (Specifies "radian" for angular unit.)

TAN (PI/4) ENTER | 1.

(Example) $\cos^{-1}$ (−Ø.5) =

(Operation) DEGREE ENTER (Specifies "degree" for angular unit.)

ACS−Ø.5 ENTER | 12Ø. |

(Example) log 5 + ln 5 =

(Operation) LOG 5 + LN 5 ENTER | 2.3Ø84Ø7917 |

(Example) e^{2+3} =

(Operation) EXP (2+3) ENTER 148.4131591 |

(Example) $\sqrt{4^3+6^4}$ =

(Operation) SQR (4 ^ 3 + 6 ^ 4) ENTER | 36.87817783 |

(Example) Convert 3Ø deg. 3Ø min. in sexagenary notation into decimal notation.

(Operation) DEG 3Ø.3Ø ENTER | 3Ø.5 |

(3Ø.5 degree)

(Example) Convert 3Ø.755 deg. in decimal notation into sexagenary notation.

(Operation) DMS 3Ø.755 ENTER | 3Ø.4518 |

(3Ø deg. 45 min. 18 sec.)

(Example) Convert CF8 to its decimal equivalent.

(Operation) &CF8 ENTER | 332Ø. |

Priority in Manual Calculation

You can type in formulas in the exact order in which they are written, including parentheses or functions. The order of priority in calculation and treatment of intermediate results will be taken care of by the computer itself.

The internal order of priority in manual calculation is as follows:

1) Recalling variables.
2) Function (sin, cos, etc.)
3) Power (^)
4) Sign (+, −)
5) Multiplication or division (*, /)
6) Addition or subtraction (+, −)
7) Comparison of magnitude (>, >=, <, <=, <>, =)
8) Logical AND, OR

Notes:
- * If parentheses are used in a formula, the operator given within the parentheses has the highest priority.
- * Composite functions are operated from right to left ($\sin \cos^{-1} 0.6$).
- * Chained power ($(3^4)^2$ or 3 ^ 4 ^ 2) are operated from right to left.
- * For the above items 3) and 4), the last entry has a higher priority.

  (e.g.) $-2 \wedge 4$ — $-(2^4)$
  $3 \wedge -2$ — 3^{-2}

Printing for Manual Calculations

The results of manual calculations are usually only displayed (on the display unit). Setting the print switch to the "P" position (print mode) prints your calculation and its result. If you do not require a print-out, set the print switch to the " . " position (non-print mode).

- ● If an error occurs during manual calculations, only the entered keys will be printed. No result will be printed. The error will be indicated on the display unit.
- ● If your entry starts with a BASIC command, nothing will be printed.
- ● When the printer is set to draw graphs, it cannot print the results of manual calculations. Set the print switch to the ' . " position. (See page 37 for details.)

Caution

Calculation Error

The following types of errors occur in ordinary calculators, pocket computers, and personal computers.

(1) Errors due to least significant digit processing

Usually, the maximum number of digits that can be calculated in a computer is fixed. For example, 4÷3 results in 1.3333333333... In a computer with a maximum of 8 digits, the 8 digits are significant digits; other least significant digits are either truncated or rounded.

(example) computer with 1Ø significant digits

$$4/3\ \boxed{\text{ENTER}} \rightarrow \overbrace{1.3333333333}^{\text{10 significant digits}}\underbrace{3}_{\text{Truncated, rounded}}\ldots$$

Therefore, the calculated result differs from the true value by the amount truncated or rounded. (This difference is the error.)
In this unit, a 12-digit calculated result is obtained. This result is rounded and specially processed to minimize error in the displayed value.

(example) $4 \div 3 \times 3$

① 4/3*3 ENTER → 4. Calculated in succession

② 4/3 ENTER → 1.333333333 ⎫
    *3 ENTER → 3.999999999 ⎭ Calculated independently

* When calculated in succession, the result of 4 ÷ 3 is obtained internally in 12 digits and is used for calculation and then rounded.
  When calculated independently, the displayed value (1Ø digits) is used for the calculation.

(2) Errors due to Approximation Calculations

Since functions are calculated using approximation algorithms, the errors generated are larger than those from regular calculations.
Although the PC-2500 performs various processes to minimize error in the displayed result, the errors become large especially near the singular point or inflection point of the function and may appear in the displayed result. Further, errors accumulate when requiring many approximation algorithms.

Example: $60^6 =$
60 ^ 6 ENTER → 4.66559999 E 1Ø

Although 60^6 equals 4.6656×10^6, the PC-2500 calculates the power function (y^x) using the following formula.

$y^x = 10^{x \log y}$

In other words, 60^6 is obtained by calculating $10^{6 \times \log 60}$.

CHAPTER 4
CONCEPTS AND TERMS OF BASIC

In this Chapter we will examine some concepts and terms of the BASIC language.

String Constants

In addition to numbers, there are many ways that the SHARP PC-2500 uses letters and special symbols. These letters, numbers, and special symbols are called characters. These characters are available on the PC-2500:

1 2 3 4 5 6 7 8 9 Ø
A B C D E F G H I J K L M N O P Q R S T U V W X Y Z
a b c d e f g h i j k l m n o p q r s t u v w x y z
! @ # $ % ^ & * ( ) / ? – _ = + ' " [ ] ; : , < . >

In BASIC, a collection of characters is called a string. In order for the PC-2500 to tell the difference between a string and other parts of a program, such as verbs or variable names, you must enclose the characters of the string in quotation marks (").

The following are examples of string constants:

"HELLO"
"Goodbye"
"SHARP PC-2500"

The following are not valid string constants:

"COMPUTER	No ending quote
"ISN"T"	Quote can't be used within a string

Hexadecimal Numbers

The decimal system is only one of many different systems to represent numbers. Another which has become quite important when using computers is the hexadecimal system. The hexadecimal system is based on 16 instead of 1Ø. To write hexadecimal numbers you use the familiar Ø ~ 9 and 6 more "digits": A, B, C, D, E, and F. These correspond to 1Ø, 11, 12, 13, 14, and 15. When you want the PC-2500 to treat a number as hexadecimal put an ampersand '&' character in front of the numeral:

&A = 1Ø
&1Ø = 16
&1ØØ = 256
&FFFF = 65535

Variables

Computers are made up of many tiny memory areas called bytes. Each byte can be thought of as a single character. For instance, the word byte requires four bytes of memory because there are four characters in it. To see how many bytes are available for use, simply type in MEM ENTER. The number displayed is the number of bytes available for writing programs. This technique works fine for words, but is very inefficient when you try to store numbers. For this reason, numbers are stored in a coded fashion. Thanks to this coding technique, your computer can store large numbers in only eight bytes. The largest number than can be stored is +9.999999999E+99.

The smallest number is +1.E–99. This gives you quite a range to choose from. However, if the result of a calculation exceeds this range, the computer will let you know by displaying an error message on the screen. For the error message refer to Appendix A. To see it right now type in:

```
9 E 99*9 ENTER
```

To get the computer working properly again, just press the CLS key. But how do you go about storing all this information? It's really very easy. The computer likes to use names for different pieces of data. Let's store the number 556 into the computer. You may call this number by any name that you wish, but for this exercise, let's use the letter R. The statement, LET, can be used to instruct the computer to assign a value to a variable name but only in a program statement. However, the LET command is not necessary, so we will not use it very often. Now, type in R=556 and press the ENTER. The computer now has the value 556 associated with the letter R. These letters that are used to store information are called variables. To see the content of the variable R, press the CLS key, the R key and the ENTER key. The computer responds by showing you the value 556 on the right of your screen. This ability can become very useful when you are writing programs and formulas.

Next, let's use the R variable in a simple formula. In this formula, the variable R stands for the radius of a circle whose area we want to find. The formula for the area of a circle is: $A=PI*R^2$. Type in R. SHIFT + ^ 2 * PI ENTER . The result is 971179.3866. This technique of using variables in equations will become more understandable as we get into writing programs.

So far, we've only discussed numeric variables. What about storing alphabetic characters? Well, the idea is the same, but, so the computer will know the difference between the two kinds of variables, add a $ to the variable name. For instance, let's store the word BYTE in the variable B$. Notice the $ after the B?

This tells the computer that the contents of the letter B is alphabetic, or string data.

To illustrate this, key in B SHIFT + $ = SHIFT + " BYTE SHIFT + " ENTER . The value BYTE is now stored in the variable B$. To make sure of this, press the CLS key and type in B SHIFT + $ ENTER . The screen shows BYTE. This time the display is on the left side of the screen, instead of the right.
The maximum number of characters that can be stored in a simple string variable is 7.

Note: The contents of character strings or character variables are displayed from the left edge of the first line.

Variables handled by the SHARP PC-2500 are divided into the following:

- Variables
  - Numeric variables
    - Fixed numeric variables (A to Z)
    - Simple numeric variables (AB, C1, etc.)
    - Numeric array variables
  - String variables
    - Fixed character variables (A$ to Z$)
    - Simple character variables (BB$, C2$, etc.)
    - Character array variables

Fixed Variables

The first section, fixed variables, is always used by the computer for storing data. It can be thought of as pre-allocated variable space. In other words, no matter how much memory your program uses up, you will always have at least 26 variables to choose from to store data in. This data can be one of two types: NUMERIC or STRING (alphabetic character). Fixed memory locations are eight bytes long and can be used for only one type of data at a time. To illustrate this, type in the following example:

```
A=123 ENTER
A$ ENTER
```

You get the message:

```
ERROR 9
```

This means that you have put numeric data into the area of memory called A and then told the computer to show you that information again as STRING data. This confuses the computer so it says that there is an error condition. Press the CLS key to clear error condition. Now try the following example:

```
A$="ABC" ENTER
A ENTER
```

Again, the computer is confused and gives the ERROR 9 message. Look at the Figure shown below to see that the variable name A equals the same area in memory as the variable name A$, and that B equals B$, and so on for all the letters of the alphabet.

A = A$ = A(1) = A$(1)

B = B$ = A(2) = A$(2)

C = C$ = A(3) = A$(3)

D = D$ = A(4) = A$(4)

E = E$ = A(5) = A$(5)

F = F$ = A(6) = A$(6)

G = G$ = A(7) = A$(7)

H = H$ = A(8) = A$(8)

I = I$ = A(9) = A$(9)

J = J$ = A(1Ø) = A$(1Ø)

K = K$ = A(11) = A$(11)

L = L$ = A(12) = A$(12)

M = M$ = A(13) = A$(13)

N = N$ = A(14) = A$(14)

O = O$ = A(15) = A$(15)

P = P$ = A(16) = A$(16)

Q = Q$ = A(17) = A$(17)

R = R$ = A(18) = A$(18)

S = S$ = A(19) = A$(19)

T = T$ = A(2Ø) = A$(2Ø)

U = U$ = A(21) = A$(21)

V = V$ = A(22) = A$(22)

W = W$ = A(23) = A$(23)

X = X$ = A(24) = A$(24)

Y = Y$ = A(25) = A$(25)

Z = Z$ = A(26) = A$(26)

Simple Variables

Simple variable names are specified by two (or more) alphanumeric characters, such as AA or B1. Unlike fixed variables, simple variables have no dedicated storage area in the memory. The area for simple variables is automatically set aside (within the program and data area) when a simple variable is first used.

Since separate memory areas are defined for simple numeric variables and simple character variables even if they have the same name, variables such as AB and AB$, for example, may be used at the same time.

While alphanumeric characters are usable for simple variable names (as for alphabetic characters, only upper case characters are usable), the first character of a variable name must always be an alphabetic character. If more than two characters are used to define a variable name, only the first two characters are meaningful.

Note:
- The function or BASIC instruction names for the PC-2500 computer are not usable for variable names. (e.g.) PI, IF, TO, ON, SIN, etc.
- Each simple character variable can hold up to 16 characters or symbols.

Array Variables

For some purposes it is useful to deal with numbers as an organized group, such as a list of scores or a tax table. In BASIC these groups are called arrays. An array can be either one-dimensional, like a list, or two-dimensional, like a table.

To define an array, the DIM (short for dimension) statement is used. Arrays must always be "declared" (defined) before they are used. (Not like the single-value variables we have been using.) The form for the numeric DIMension statement is:

DIM numeric-variable-name (size)

where:

numeric-variable-name is a variable name which conforms to the normal rules for numeric variable names perviously discussed.

size is the number of storage locations and must be a number in the range Ø through 255. Note that when you specify a number for the size, you get one more location than you specified.

Examples of legal numeric DIMension statements are:

```
    DIM X (5)
    DIM AA (24)
    DIM Q5 (Ø)
```

The first statement creates an array X with 6 storage locations. The second statement creates an array AA with 25 locations. The third statement creates an array with one location and is actually rather silly since (for numbers at least), it is the same as declaring a single-value numeric variable.

It is important to know that an array-variable X and a variable X are separate and distinct to the PC-2500. The first X denotes a series of numeric storage locations, and the second a single and different location.

Now that you know how to create arrays, you might be wondering how it is that we refer to each storage location. Since the entire group has only one name, the way in which we refer to a single location (called an "element") is to follow the group name with a number in parentheses. This number is called a "subscript". Thus, for example, to store the number 8 into the fifth element of our array X (declared previously) we would write:

```
X(4)=8
```

If the use of 4 is puzzling, remember that the numbering of elements begins at zero and continues through to the number declared in the DIM statement.

The real power of arrays lies in the ability to use an expression or a variable name as a subscript.

To declare a character array a slightly different form of the DIM statement is used:

DIM character-variable-name (size) * length

where:

character-variable-name is a variable name which conforms to the rules for normal character variables as discussed previously.

size is the number of storage locations and must be in the range Ø through 255. Note that when you specify a number, you get one more location than you specified.

*length is optional. If used, it specifies the length of the strings that comprise the array. Length is a number in the range 1 to 8Ø. If this clause is not used, the strings will have the default length of 16 characters.

Example of legal character array declarations are:

```
    DIM  X$(4)
    DIM  NM$(1Ø)*1Ø
    DIM  IN$(1)*8Ø
    DIM  R$(Ø)*26
```

The first example creates an array of five strings each able to store 16 characters. The second DIM statement declares an array NM with eleven strings of 1Ø characters each.

Explicit definition of strings smaller than the default helps to conserve memory space. The third example declares a two element array of 8Ø-character strings and the last example declares a single string of twenty-six characters.

Besides the simple arrays we have just studied, the PC-2500 allows "two-dimensional" arrays. By analogy, a one-dimensional array is a list of data arranged in a single column. A two-dimensional array is a table of data with rows and columns.

The two-dimensional array is declared by the statement:

DIM numeric-variable-name (rows, columns)

or

DIM character-variable-name (rows, columns) * length

where:

row specifies the number of rows in the array. This must be a number in the range Ø through 255. Note that when you specify the number of rows you get one more row than the specification.

columns specifies the number of columns in the array. This must be a number in the range Ø through 255. Note that when you specify the number of columns you get one more column than the specification.

The following diagram illustrates the storage locations that result from the declaration DIM T (2, 3) and the subscripts (now composed of two numbers) which pertain to each storage location:

	column Ø	column 1	column 2	column 3
row Ø	T(Ø, Ø)	T(Ø, 1)	T(Ø, 2)	T(Ø, 3)
row 1	T(1, Ø)	T(1, 1)	T(1, 2)	T(1, 3)
row 2	T(2, Ø)	T(2, 1)	T(2, 2)	T(2, 3)

Note: Two-dimensional arrays can rapidly eat up storage space. For example, an array with 25 rows and 35 columns uses 875 storage locations!

Arrays are very powerful programming tools.

The following table shows the number of bytes used to define each variable and the number used by each program statement.

Variable	Variable name	Data	
Numeric variable	7 bytes	8 bytes	
String variable	7 bytes	Array variable	Specified number*
		Simple variable (two-character variable)	16 bytes

* For example, if DIM Z$(2, 3)*1Ø is specified, 12 variables, each capable of storing 1Ø characters, are reserved. This requires 7 bytes (variable name) + 1Ø bytes (number of characters) x 12 = 127 bytes.

Element	Line number	Statement & function	Special symbols*	ENTER , and others
Number of bytes used	3 bytes	1 bytes	2 bytes	1 bytes

* Ex. character code 245 to 252

Variables in the Form of A( )

While a data area on the computer's memory is set aside for fixed variables, it may also be used to define subscripted variables which have the same form as array variables.

There are 26 fixed variable names available: i.e. A through Z(A$ through Z$). Each of these names can be subscripted with the numbers 1 through 26, such as A(1)–A(26) or A$(1)–A$(26). This means that variable A(1) may be used in place of variable A, A(2) in place of B, A(3) in place of C, and so forth.

However, if an array named A or A$ has already been defined by the DIM statement, subscripted variables named A cannot be defined. For example, if an array A is defined by DIM A(5) the locations for A(Ø) through A(5) are set aside in the program/data area. So if you specify variable A(2), it does not refer to the fixed variable B, but refers to the array variable A(2) defined in the program/data area. If you specify A(9), it will cause an error since A(9) is outside the range of the dimension specified by the DIM A(5) statement.

On the other hand, if subscripted variables are already defined in the form of A( ), it is not possible to define arrays A or A$ by using the DIM statement, unless the definition for the subscripted variables is cleared with the CLEAR statement.

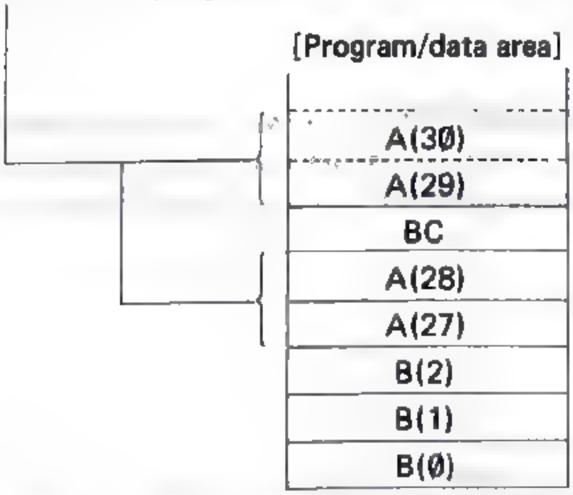
* Using subscripts in excess of 26:
If subscripts greater than 26 are used for subscripted variables A( ) when array A is not defined by a DIM statement, the corresponding locations in the program/data area are set aside for these A( ) variables. For instance, if you execute A(35)=5, locations for variables A(27) to A(35) will be reserved in the program/data area.

While variables subscripted in excess of 26 are treated as array variables, they are subject to the following special restrictions;

(1) Locations for an array with the same name must be contiguous in the program/data area. Otherwise, an error will occur.

```
1Ø DIM B(2)
2Ø A(28)=5
3Ø BC=12
4Ø A(3Ø)=9
```

If this program is executed, the array named "A" is not defined in two consecutive segments in the program data area, and an error will result at line 4Ø.



(2) Numeric array variables and character array variables with the same subscript cannot be defined at the same time. For example, A(3∅) and A$(3∅) cannot be defined at the same time, since they use the same location in the program/data area.

(3) Two dimensional arrays cannot be defined, nor is it possible to specify the length of character strings to be held in character array variables. For example, the length of a character string which can be held in the character array variable A$( ) is limited to seven characters or less.

(4) Variables subscripted with zero (∅) cannot be defined. If A(∅) or A$(∅) is defined, an error will result.

(5) When A(27), or A$(27) and higher is first used, 7 bytes are used for the variable names and 8 bytes are used for each variable.

Expressions

An expression is some combination of variables, constants, and operators which can be evaluated to a single value. The calculations which you entered in Chapter 3 were examples of expressions. Expressions are an intrinsic part of BASIC programs. For example, an expression might be a formula that computes an answer to some equation, a test to determine the relationship between two quantities, or a means to format a set of strings.

Numeric Operators

The PC-2500 has five numeric operators. These are the arithmetic operators which you used when exploring the use of the PC-2500 as a calculator in Chapter 3:

+	Addition
−	Subtraction
*	Multiplication
/	Division
^	Exponent

A numeric expression is constructed in the same way that you entered compound calculator operations. Numeric expressions can contain any meaningful combination of numeric constants, numeric variables, and the numeric operators:

```
(A*B)^2
B(2, 3) + B(3, 4) + 5.∅−C
(A/B)*(C+D)
```

String Expressions

String expressions are similar to numeric expressions except that there is only one string operator—concatenation (+). This is the same symbol used for plus. When used with a pair of strings, the + attaches the second string to the end of the first string and makes one longer string. You should take care in making more complex string concatenations and other string operations because the work space used by the PC-2500 for string calculations is limited to only 8Ø characters.

Note: String quantities and numeric quantities cannot be combined in the same expression unless one uses one of the functions which convert a string value into a numeric value or vice versa:

```
"15" + 1Ø       is illegal
"15" + "1Ø"     is "151Ø", not "25"
```

Relational Expressions

A relational expression compares two expressions and determines whether the stated relationship is true or false. The relational operators are:

>	Greater Than
>=	Greater Than or Equal To
=	Equals
<>	Not Equal To
<=	Less Than or Equal To
<	Less Than

The following are valid relational expressions:

```
A<B
C(1,2)>=5
D(3)<>8
```

If A was equal to 1Ø, B equal to 12, C(1, 2) equal to 6, and D(3) equal to 9, all of these relational expression would be true.

Character strings can also be compared in relational expressions. The two strings are compared character by character according to their ASCII value starting at the first character (see Appendix B for ACII values). If one string is shorter than the other, a Ø or NUL will be used for any missing positions. All of the following relational expressions are true:

```
"ABCDEF" = "ABCDEF"
"ABCDEF" <> "ABCDE"
"ABCDEF" > "ABCDE"
```

Relational expressions evaluate to either true or false. The PC-2500 represents true by a 1; false is represented by a Ø. In any logical test an expression which evaluates to 1 or more will be regarded as true while one which evaluates to Ø or less will be considered false. Good programming practice, however, dictates the use of an explicit relational expression instead of relying on this coincidence.

Logical Expressions

Logical expressions are relational expressions which use the operators AND, OR, and NOT. AND and OR are used to connect two relational expressions; the value of the combined expression is shown in the following tables:

A AND B

		Value of A	
		True	False
Value of B	True	True	False
	False	False	False

A OR B

		Value of A	
		True	False
Value of B	True	True	True
	False	True	False

(Note: Value of A and B must be Ø or 1)

- Decimal numbers can be expressed in the binary notation of 16 bits as follows:

DECIMAL NOTATION	BINARY NOTATION OF 16-BIT
32767	Ø111111111111111
⋮	
3	ØØØØØØØØØØØØØØ11
2	ØØØØØØØØØØØØØØ1Ø
1	ØØØØØØØØØØØØØØØ1
Ø	ØØØØØØØØØØØØØØØØ
–1	1111111111111111
–2	111111111111111Ø
–3	11111111111111Ø1
⋮	⋮
–32768	1ØØØØØØØØØØØØØØØ

The negative (NOT) of a binary number ØØØØØØØØØØØØØØØ1 is taken as follows:

```
NOT                ØØØØØØØØØØØØØØØ1
(Negative)  →      111111111111111Ø
```

Thus, 1 is inverted to Ø, and Ø to 1 for each bit, which is called "to take negative (NOT)".

Then, the following will result when 1 and NOT 1 are added together:

```
    ØØØØØØØØØØØØØØØ1 (1)
+)  111111111111111Ø (NOT 1)
    1111111111111111 (−1)
```

Thus, all bits become 1. According to the above number list, the bits become −1 in decimal notation, that is 1 + NOT 1 = −1.

The relationship between numerical value X and its negative (NOT X) is:

X + NOT X = −1

This results in an equation of NOT X = −X−1

i.e. NOT X = −(X + 1)

From the equation the following are found to result.

NOT Ø = −1

NOT −1 = Ø

NOT −2 = 1

More than two relational expressions can be combined with these operators. You should take care to use parentheses to make the intended comparison clear.

(A<9) AND (B>5)

(C=5) OR (C=6) OR (C=7)

The PC-2500 implements logical operators as "bitwise" logical functions on 16 bit quantities. (See note on relational expressions and true and false). In normal operations this is not significant because the simple 1 and Ø (true and false) which result from a relational expression uses only a single bit. If you apply a logical operator to a value other than Ø or 1, it works on each bit independently. For example if A is 17, and B is 22, (A OR B) is 23:

17 in binary notation is	1ØØØ1
22 in binary notation is	1Ø11Ø
17 OR 22 is	1Ø111 (1 if 1 in either number, otherwise Ø)

1Ø111 is 23 in decimal.

If you are a proficient programmer, there are certain applications where this type of operation can be very useful. Beginning programmers should stick to clear, simple true or false relational expressions.

Parentheses and Operator Precedence

When evaluating complex expressions the PC-2500 follows a predefined set of priorities which determine the sequence in which operators are evaluated. This can be quite significant:

5+2*3 could be

5+2=7 7*3=21	or	2*3=6 6+5=11

The exact rules of "operator precedence" are given in Appendix D.

To avoid having to remember all these rules and to make your program clearer, always use parentheses to determine the sequence of evaluation. The above example is clarified by writing either:

(5+2)*3 or 5+(2*3)

RUN Mode

In general, any of the above expressions can be used in the RUN mode as well as in programming a BASIC statement. In the RUN mode an expression is computed and displayed immediately. For example:

Input	Display
(5>3) AND (2<6)	1.

The 1 means that the expression is true.

Functions

Functions are special components of the BASIC language which take one value and transform it into another value. Functions act like variables whose value is determined by the value of other variables or expressions. ABS is a function which produces the absolute value of its argument:

ABS(−5)	is 5
ABS(6)	is 6

LOG is a function which computes the log to the base 1Ø of its argument.

LOG (1ØØ)	is	2
LOG (1ØØØ)	is	3

A function can be used any place that a variable can be used. Many functions do not require the use of parentheses:

LOG 1ØØ is the same as LOG (1ØØ)

You must use parentheses for functions which have more than one argument. Using parentheses always makes programs clearer.

See Chapter 8 for a complete list of functions available on the PC-2500.

CHAPTER 5
PROGRAMMING THE PC-2500

In the previous chapter we examined some of the concepts and terms of the BASIC programming language. In this chapter you will use these elements to create programs on the PC-2500. Let us reiterate however, this is not a manual on how to program in BASIC. What the chapter will do is familiarize you with the use of BASIC on your PC-2500.

Programs

A program consists of a set of instructions to the computer. Remember the PC-2500 is only a machine. It will perform the exact operations that you specify. You, the programmer, are responsible for issuing the correct instructions.

BASIC Statements

The PC-2500 interprets instructions according to a predetermined format. This format is called a statement. You always enter BASIC statements in the same pattern. Statements must start with a line number:

```
1Ø: INPUT A
2Ø: PRINT A*A
3Ø: END
```

Line Numbers

Each line of a program must have a unique line number—any integer between 1 and 65279. Line numbers are the reference for the computer. They tell the PC-2500 the order in which to perform the program. You need not enter lines in sequential order (although if you are a beginning programmer, it is probably less confusing for you to do so). The computer always begins execution with the lowest line number and moves sequentially through the lines of a program in ascending order.

When programming it is wise to allow increments in your line numbering (1Ø, 2Ø, 3Ø, . . . 1Ø, 3Ø, 5Ø etc). This enables you to insert additional lines if necessary.

CAUTION: Do not use the same line numbers in different programs you plan to merge.

If you use the same line number, the oldest line with that number is deleted when you enter the new line.

BASIC Verbs

All BASIC statements must contain verbs. Verbs tell the computer what action to perform. A verb is always contained within a program, and as such is not acted upon immediately.

Some statements require or allow an operand:

```
10: DATA "HELLO"
20: READ B$
30: PRINT B$
40: END
```

Operands provide information to the computer telling it what data the verb will act upon. Some verbs require operands, with other verbs they are optional. Certain verbs do not allow operands. (See Chapter 9 for a complete listing of BASIC verbs and their use on the PC-2500.)

BASIC Commands

Commands are instructions to the computer which are entered outside of a program. Commands instruct the computer to perform some action with your program or to set modes which affect how your programs are executed.

Unlike verbs, commands have immediate effects—as soon as you complete entering the command (by pressing the ENTER key), the command will be executed. Commands are not preceded by a line number:

```
RUN
NEW
RADIAN
```

Some verbs may also be used as commands. (See Chapter 9 for a complete listing of BASIC commands and their use on the PC-2500).

Modes

You will remember that when using the PC-2500 as a calculator, it is set in the RUN mode.

The RUN mode is also used to execute the programs you create.

The PROgram mode is used to enter and edit your programs.

The RESERVE mode enables you to designate and store predefined string variables and is used in more advanced programming (See Chapter 6).

Beginning to Program on the PC-2500

After all your practice in using the PC-2500 as a calculator you are probably quite at home with the keyboard. From now on, when we show an entry, we will not show every keystroke. Remember to use SHIFT to access characters on the upper half of the keys and END EVERY LINE BY PRESSING THE ENTER KEY.

Now you are ready to program!

To enter program statements into the computer, the computer must first be placed in the PROGRAM mode using the MODE key. The display will appear as in the following illustration.

```
PROGRAM MODE              PRO
>
```

Enter NEW command.

Input	Display
NEW	>

The NEW command clears the PC-2500 memory of all existing programs and data. The prompt appears after you press ENTER , indicating that the computer is awaiting input.

Example 1 — Entering and Running a Program

Make sure the PC-2500 is in the PRO mode and enter the following program:

Input	Display
1Ø PRINT "HELLO"	1Ø: PRINT "HELLO"

Notice that when you push ENTER the PC-2500 displays your input, automatically inserting a colon (:) between the line number and the verb. Verify that the statement is in the correct format.

Now change the mode to RUN:

Input	Display
RUN	HELLO

Since this is the only line of the program, the computer will stop executing at this point. Press ENTER to get out of the program and reenter RUN if you wish to execute the program again.

Example 2 — Editing a Program

Suppose you wanted to change the message that your program was displaying, that is, you wanted to edit your program. With a single line program you could just retype the entry, but as you develop more complex programs editing becomes a very important component of your programming. Let's edit the program you have just written.

Are you still in the RUN mode? If so switch back to the PROgram mode.

You need to recall your program in order to edit it. Use the Up Arrow [↑] to recall your program. If your program was completely executed, the [↑] will recall the last line of the program. If there was an error in the program, or if you used the BREAK ([illegible]) key to stop execution, the [↑] will recall the line in which the error or BREAK occurred. To make changes in your program use the [↑] to move up in your program (recall the previous line) and the [↓] to move down in your program (display the next line). If held down, the [↑] and the [↓] will scroll vertically, that is, they will display each line moving up or down in your program.

You will remember that to move the cursor within the program line displayed at the top line of the display you use the [→] (right arrow) and [←] (left arrow). Using the [→] position the cursor over the first character you wish to change:

Note: Even if several lines of a program are displayed on the display unit, the cursor can be moved only within the first displayed line. To edit a lower line, move the line to the top using the [↓] key and then edit.

Input	Display
[↑]	1Ø: PRINT "HELLO█
[←] [←] [←] [←] [←]	1Ø PRINT "█ELLO"

Notice that the cursor is now in the flashing block form indicating that it is "on top of" an existing character. Type in:

Input	Display
GOODBYE"!	1Ø PRINT "GOODBYE"!_

Don't forget to press [ENTER] at the end of the line. Change the mode to RUN.

Input	Display
RUN [ENTER]	ERROR 1 IN 10

This is a new kind of error message. Not only is the error type identified (our old friend the syntax error) but the line number in which the error occurs is also indicated.

Press the CLS to clear the error condition, then return to the PRO mode. You must be in the PROgram mode to make changes in a program. Using the ↑ (or ↓), recall the line in which the error occurred.

Input	Display
↑ (or ↓)	1Ø: PRINT "GOODBYE"█

The flashing cursor is positioned over the problem area. In Chapter 4 you learned that when entering string constants in BASIC all characters must be contained within quotation marks. Use the DELete key while holding down the SHIFT key to eliminate the "!":

Input	Display
DEL	1Ø PRINT "GOODBYE"_

Now let's put the ! in the correct location. When editing programs, DELete and INSert are used in exactly the same way as they are in editing calculations (See Chapter 3). Using the ← position the cursor on top of the character which will be the first character following the insertion.

Input	Display
←	1Ø PRINT "GOODBYE█

Press the INSert key. A ⊏ will indicate the spot where the new data will be entered:

Input	Display
INS	1Ø PRINT "GOODBYE ⊏"

Type in the !. The display looks like this:

Input	Display
!	1Ø PRINT "GOODBYE!"

Remember to press ENTER so the correction will be entered into the program.

Note: If you wish to DELete an entire line from your program just type in the line number and the original line will be eliminated.

Example 3 – Using Variables in Programming

If you are unfamiliar with the use of numeric and string variables in BASIC, reread these sections in Chapter 4.

Using variables in programming allows much more sophisticated use of the PC-2500's computing abilities.

Remember, you assign simple numeric variables using any letter from A to Z:

```
A=5
```

To assign string variables you also use a letter, followed by a dollar sign. Do not use the same letter in designating a numeric and a string variable. You cannot designate A and A$ in the same program.

Remember that simple string variables cannot exceed 7 characters in length:

```
A$="TOTAL"
```

The values assigned to a variable can change during the execution of a program, taking on the values typed in or computed during the program. One way to assign a variable is to use the INPUT verb. In the following program the value of A$ will change in response to the data typed in answering the inquiry "WORD?". Enter this program:

```
1Ø INPUT "WORD?"; A$
2Ø B=LEN (A$)
3Ø PRINT "WORD␣IS␣"; B; "␣LETTERS"
4Ø END
```

␣ means space

Since line 3Ø of this program is longer than 24 columns, the remaining part is displayed in the next line.

The second new element in this program is the use of the END statement to signal the completion of a program. END tells the computer that the program is completed. It is always good programming practice to use an END statement.

As your programs get more complex you may wish to review them before you begin execution. To look at your program, use the LIST command. LIST, which can only be used in the PROgram mode, displays the program beginning with the lowest line number.

Try listing this program:

Input	Display
LIST	1Ø: INPUT "WORD?"; A$ 2Ø: B= LEN (A$) 3Ø: PRINT "WORD IS ";B;" LETTERS"

Use the ↑ and ↓ arrows to move through your program until you have reviewed the entire program. To review a line which contains more characters than can be seen at one time move the cursor to the extreme right of the display and the additional characters will appear on the screen. After checking your program, run it:

Input	Display
RUN	RUN WORD?_
HELP	RUN WORD? HELP_
ENTER	RUN WORD? HELP WORD IS 4. LETTERS

This is the end of your program. Of course you may begin it again by entering RUN. However, this program would be a bit more entertaining if it presented more than one opportunity for input. We will now modify the program so it will keep running without entering RUN after each answer.

Return to the PRO mode and use the up or down arrow (or LIST) to reach line 4Ø. Press the up or down arrow key until the Line 4Ø comes to the top of the screen. (or type LIST 4Ø and Press ENTER)

Display
4Ø: END

You may type 4Ø to Delete the entire line or use the → to position the cursor over the E in End. Change line 4Ø so that it reads:

```
4Ø:  GOTO 1Ø
```

Now RUN the modified program.

The GOTO statement causes the program to loop (keep repeating the same operation). Since you put no limit on the loop it will keep going forever (an "infinite" loop). To stop this program hit the BREAK (ON) key.

When you have stopped a program using the ON key, you can restart it using the CONT command. CONT stands for CONTinue. With the CONT command the program will restart on the line which was being executed when the ON key was pressed.

Example 4 — More Complex Programming

The following program computes N factorial (N!). The program begins with 1 and computes N! up to the limit which you enter. Enter this program.

```
100 F=1: WAIT 118
110 INPUT "LIMIT?"; L
120 FOR N=1 TO L
130 F=F*N
140 PRINT N, F
150 NEXT N
160 END
```

Several new features are contained in this program. The WAIT verb in line 100 controls the length of time that displays are held before the program continues. The numbers and their factorials are displayed as they are computed. The time they appear on the display is set by the WAIT statement to approximately 2 seconds, instead of waiting for you to press ENTER .

Also in line 100, notice that there are two statements on the same line separated by a colon (:). You may put as many statements as you wish on one line, separating each by a colon, up to the 80 character maximum including ENTER . Multiple statement lines can make a program hard to read and modify, however, so it is good programming practice to use them only where the statements are very simple or there is some special reason to want the statements on one line.

Also in this program we have used the FOR verb in line 120 and the NEXT verb in line 150 to create a loop. In Example 3 you created an "infinite" loop which kept repeating the statements inside the loop until you press the ON key. With this FOR/NEXT loop the PC-2500 adds 1 to N each time execution reaches the NEXT verb. It then tests to see if N is larger than the limit L. If N is less than or equal to L, execution returns to the top of the loop and the statements are executed again. If N is greater than L, execution continues with line 160 and the program stops.

You may use any numeric variable in a FOR/NEXT loop. You also do not have to start counting at 1 and you can add any amount at each step. See Chapter 9 for details.

We have labelled this program with line numbers starting with 100. Labelling programs with different line numbers allows you to have several programs in memory at one time. To RUN this program instead of the one at line 1Ø enter:

```
    RUN 1ØØ
```

In addition to executing different programs by giving their starting line number, you can give programs a letter name and start them with the DEF key (see Chapter 6).

Storing Programs in the PC-2500 Memory

You will remember that settings, ReSerVe keys, and functions remain in the computer even after it is turned OFF. Programs also remain in memory when you turn off the PC-2500, or it undergoes an AUTO OFF. Even if you use the ON/BRK , CLear or CA keys the programs will remain.

Programs are lost from memory only when you perform the following actions:

- You enter NEW before beginning programming.
- You initialize the computer using the ALL RESET button.
- You create a new program using the SAME LINE NUMBERS as a program already in memory.
- You change the batteries.

This brief introduction to programming on the PC-2500 should serve to illustrate the exciting programming possibilities of your new computer. For more practice in programming exercises, please see Program examples.

Graphic Functions

The display unit (screen) of the PC-2500 is composed of 150 horizontal and 32 vertical dots (points). Simple pictures can be drawn on the screen using these dots. The following 6 commands are available for drawing pictures,

GPRINT: Graphic PRINT. This command is used to produce patterns with 8 vertical dots per unit.

GCURSOR: Graphic CURSOR. This command is used to specify the position when drawing a pattern with GPRINT.

PSET: Point SET. This command is used to light up or reverse a specified point (dot).

PRESET: Point RESET. This command clears the specified point (dot).

LINE: This command is used to draw a line or a square between 2 specified points.

POINT: This command is used to determine whether the specified dot is lit or not.

The basics of picture drawing will be described here. For details on the function of each command, see the description for each command.

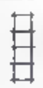
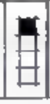
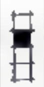
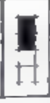
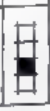
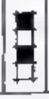
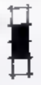
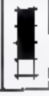
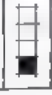
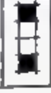
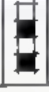
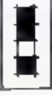
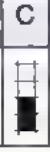
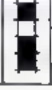
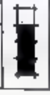
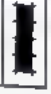
Basically, there are 2 ways to draw pictures.

The first is to draw pictures by combining predetermined patterns. The other is to draw pictures by specifying and lighting up each dot (point) as necessary.

To display a combination of characters and numeric values along with graphics, use the graphic command after displaying the characters and numeric values.

(1) Drawing a Picture Using Predetermined Patterns

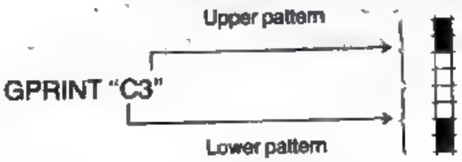
A picture is drawn by combining the 16 available patterns shown in the table below and using the GPRINT command.

Hexadecimal character	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Pattern																

The patterns in this table all use 4 dots (points).

However, in the GPRINT command, two patterns are combined vertically for an 8-dot pattern as shown in the example below. A picture is drawn by combining and lining up a number of 8-dot patterns.

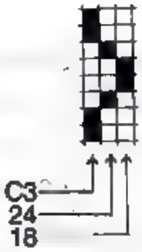
(Example)



A pair of hexadecimal numbers are used with the first specifying the lower pattern and the second specifying the upper pattern.

(Example)

GPRINT "C32418"

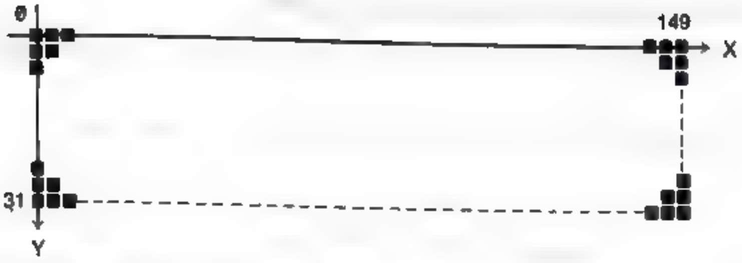


The 8-dot patterns specified by the hexadecimal number pairs are arranged to draw the picture.

(2) Specifying a Location on the Screen

As described, patterns are specified using the GPRINT command. The location where the pattern is to be displayed can be specified using the GCURSOR command. The display unit (graphic screen) of the PC-2500 is composed of 150 x 32 dots (points).

Each dot, like a point on an X-Y coordinate system, can be specified in the form of (x, y), where "x" is the horizontal direction and "y" is the vertical direction.



Note, however, that in normal X-Y coordinate systems, the point is higher with larger values of "y," while in the PC-2500, the point is lower on the screen.
The coordinates of the dots on the screen range 0–149 for "x" and 0–31 for "y."

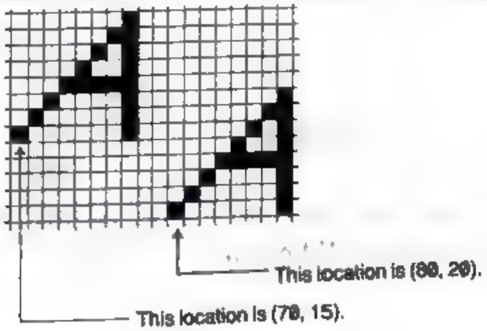
Note: The values for "x" and "y" can be specified in the range of –32768 to +32767. A virtual location (location which is made to temporarily exist but does not actually exist) will be specified if a location beyond the boundaries of the screen is specified.
Therefore, if a picture is drawn from such a location, nothing may be displayed on the screen.
This also applies to other commands described later where the coordinates are specified with such as PSET, PRESET, and LINE commands.

The GPRINT command draws pictures by lining up groups of 8-dot patterns. If the location (dot) where the picture is to be displayed was specified with the GCURSOR command, the first 8-dot pattern is displayed on the 8 dots above and including the dot at the specified location. The rest of the pattern is drawn in sequence.

(Example)

```
 5 CLS
10 AA$="8040201018141 2FF"
20 GCURSOR (70, 15)
30 GPRINT AA$
40 GCURSOR (80, 20)
50 GPRINT AA$
```

* Executing the program will display the following pattern near the center of the screen.



(3) Drawing a Picture by Specifying Dots One-by-One

A picture is drawn by lighting up or clearing the specified dot using the PSET or PRESET command.
Each dot is specified in the same way as in the GCURSOR command.

(Example)

PSET (75, 15)	Lights up the dot at (75, 15)
PSET (75, 15), X	Lights up the dot at (75, 15) if cleared, and clears it if lit. (Specifying "X" reverses the dot.)
PRESET (75, 15)	Clears the dot at (75, 15).

(Example)

```
    1ØØ WAIT Ø : DEGREE
    11Ø FOR  A=18Ø  TO  -18Ø
        STEP  -2
    12Ø B=SIN  A*12               }
    13Ø C=COS  A*12               } *1
    14Ø X=B+8Ø                    }
    15Ø Y=C+14                    } *2
    16Ø PSET (X, Y)
    17Ø NEXT  A                 ← *3
    18Ø WAIT : GPRINT           ← *4
```

*1 Determines the coordinates of the perimeter of a circle with radius 12. The center is at (Ø, Ø).
*2 Moves the center of the circle to (8Ø, 14).
*3 Lights up the dot specified by (x, y).
*4 Continues to display the drawn circle.

Executing this program draws a circle with radius 12 and center at (8Ø, 14).
The angle is changed in 2 units from +18Ø to −18Ø° in the FOR-NEXT loop. At each angle, the coordinates on the circle's perimeter are determined and the corresponding dot is lit.

(4) Drawing Lines and Squares

Although lines and squares can also be drawn using the PSET command, they can be easily drawn simply by specifying 2 dots in the LINE command.

(Example)

LINE (Ø, Ø) – (149, 31)
Draws a line from (Ø, Ø) to (149, 31).

LINE (3Ø, Ø) – (8Ø, 31), B
Draws a square with its diagonal from (3Ø, Ø) to (8Ø, 31). If B is changed to BF, the inside of the square will be filled.

```
LINE (30, 0) – (80, 31), X, BF
```

A filled box is drawn. However, if a dot within the square is already lit, it is cleared.
(Specifying "X" reverses the dots.)

(Example)

```
200 "A" : WAIT 0
210 LINE (60, 0) – (100, 31)
    , X, BF
220 GOTO 210
```

Executing this program draws a square with its diagonal from (60, 0) to (100, 31) and fills and clears it. With the picture previously drawn using the GPRINT and PSET commands still displayed, try executing the program by pressing DEF and A . The picture within the square will reverse.

Note: The graphic screen is 6 dots wider on the left side than the usual character and numeric display.
The PC-2500 is designed to retain the drawn image unless cleared. Consequently, if the program ends or is stopped, the graphic image may remain on the left side or top of the screen.
(The display is cleared by Pressing CLS .)

Serial I/O Function

The PC-2500 is equipped with a serial I/O interface. This function can be used to connect with a personal computer for data I/O.

Note 1: When connecting the RS-232C interface which is equipped on many computers to the PC-2500, an optional level shifter becomes necessary since the voltage levels of the RS-232C signals differ. A connecting cable is also necessary.

Note 2: Exercise care since applying a voltage exceeding the allowable range of the PC-2500 to the I/O terminal may damage the internal parts.

Basics on the Use of the Serial I/O Interface

The circuit of the serial I/O interface is usually closed. If closed, data from the serial I/O terminal cannot be sent and the received data cannot be read.
Therefore, it is first necessary to open the circuit. The circuit is opened using the OPEN command. (If already open, ERROR 8 results.)

Further, the conditions to perform data I/O with the computer connected to the PC-2500 must be satisfied. In other words, the conditions for the signals must be the same for both the PC-2500 and the connected personal computer. If the conditions are different, the signals (data) cannot be read correctly resulting in data errors. The OPEN command can be used to set and modify the I/O conditions.

After the conditions for both sides are satisfied and the circuit opened, the following commands are used to perform data or program I/O.

LPRINT, LLIST, SAVE, LOAD, PRINT #1, INPUT #1

After the end of the data or program I/O, the circuit of the serial I/O interface is closed. Although the CLOSE command is used to close the circuit, the circuit also closes when the program ends (such as when the END command is executed) or when the RUN command is executed.
When writing a program which uses the serial I/O interface, the circuit must be opened, the I/O operation performed, and then the circuit must be closed as described above.

Note: The PC-2500 is not designed to wait during the I/O commands to the serial I/O interface and is not equipped with a timer function to interrupt communication with the connected equipment.
Therefore, if the connected equipment is not ready to communicate (such as when the power is off) while commands are being executed or if communication at the connected equipment is interrupted, the PC-2500 cannot stop the executed commands and will continue execution.
If this is the case, press the ON/BRK key and stop the execution.

CHAPTER 6
SHORTCUTS

The PC-2500 includes several features which make programming more convenient by reducing the number of keystrokes required to enter repetitive material.

One such feature is in the availability of abbreviations for verbs and commands (See Chapter 9).

This chapter discusses two additional features which can eliminate unnecessary typing—the DEF key and the Reserve mode.

The DEF Key and Labelled Programs

Often you will want to store several different programs in the PC-2500's memory at one time. (Remember that each must have unique line numbers). Normally, to start a program with a RUN or GOTO command, you need to remember the beginning line number of each program (see Chapter 9). But, there is an easier way! You can label each program with a letter and execute the program using only two keystrokes. This is how to label a program and execute it using DEF:

Note: Put a label on the first line of each program that you want to reference. The label consists of a single character in quotes, followed by a colon:

```
1Ø: "A": PRINT "FIRST"
2Ø: END
8Ø: "B": PRINT "SECOND"
9Ø: END
```

Any one of the following characters can be used: A, S, D, F, G, H, J, K, L, Z, X, C, V, B, N, M, and SPACE.

Note: To execute the program, instead of typing RUN 8Ø or GOTO 1Ø, you need only press the DEF key and then the letter used as a label. In the above example, pressing DEF and then 'B' would cause 'SECOND' to appear on the display.

When DEF is used to execute a program, variables and mode settings are affected in the same way as when GOTO is used. See Chapter 8 for details.

RESERVE Mode

Another timesaving feature of the PC-2500 is the RESERVE mode.

Within the memory of the PC-2500, 79 characters are designated for "Reserve Memory". You can use this memory to store frequently used expressions, which are then recalled by a simple two keystroke operation.

Note:

- You store the strings in the RESERVE mode and recall them for use in the RUN and PROgram modes.
- The PC-2500 has a reserve memory of 79 bytes. Set up to 79 bytes, including the reserve key, in the reserve memory. A BASIC command, function, number, or a alphabetic character is 1 byte.

  Example: Ø: 1: + – 1 2 A B SIN COS INPUT RUN.....
  (1 byte each)
- The length of reserved string for one key is a maximum of up to 8Ø key commands, including the reserve key and ENTER key.

Try this example of storing and recalling a reserved string.

Change the PC-2500 into RESERVE mode by pressing the SHIFT + MODE keys. Notice that the mode indicator "RUN" and "PRO" disappear and the message "RESERVE MODE" is displayed when the reserve mode is set.

The NEW followed by the ENTER key. This will clear out any previously stored characters in the same way NEW clears out stored programs in the PROgram mode.

The DEF followed by '1':

Input	Display
DEF 1	1:_

Notice that the '1' appears in the display at the left followed by a colon.

Enter the word 'PRINT' and press the ENTER key:

Input	Display
PRINT ENTER	1: PRINT

A space appears after the colon signalling you that 'PRINT' is now stored in the reserve memory under the letter 1.

Switch the PC-2500 into PROgram mode. Type NEW followed by ENTER to clear the program memory. Type '1Ø' as a line number and then press DEF and the '1' key:

Input	Display
1Ø DEF 1	1Ø PRINT _
ENTER	1Ø: PRINT

Immediately the word 'PRINT' will appear in the display after the line number.

Any character sequence can be stored in reserve Memory. The stored strings can be recalled at any time in either the PROgram or the RUN mode by typing DEF and the key that the string is stored under. Key which can enter numbers 0 through 9 can be used.

To edit a stored character sequence, switch into the reserve mode and press DEF and the key under which the sequence is stored. You can then edit using the Left Arrow, Right Arrow, DEL, and INS keys in the same way as in other modes.

When the last character in a stored sequence is a '@' character, it is interpreted as ENTER when the sequence is recalled. For example, if you store the string "GOTO 1ØØ@" under the '9' key, typing DEF and '9' in the RUN mode immediately starts execution of the program at line 1ØØ. Without the '@' character, you must press ENTER after the DEF and '9' to begin execution.

To delete reserve programs:

1. As you know, NEW ENTER keys clear all reserve memories.
   Please note that the above key operation must be done in the RESERVE mode.
2. To delete a reserve memory, use the □ (space) or SHIFT + DEF or BS key as described below:

   Example: Clear A*A which is reserved in the key number 8 .

Input	Display	Remarks
DEF 8	8: _	Reserve mode
A*A ENTER	8: A*A	
CLS	>	
DEF 8	8: A*A	
← or →	8: A*A	
SHIFT + DEL . . . SHIFT + DEL	8: _	Delete A*A
ENTER	>	

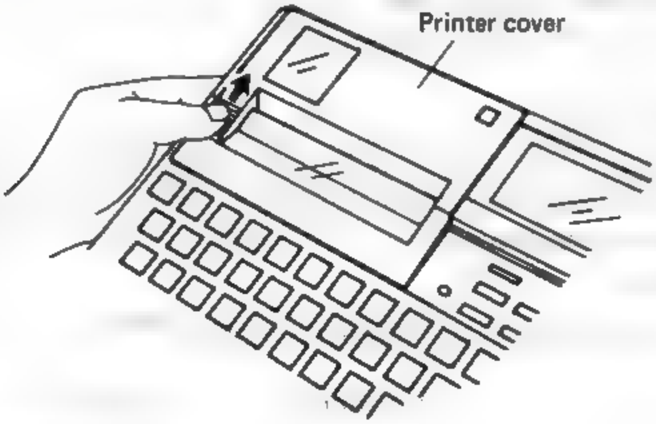
CHAPTER 7 USING THE PRINTER/CASSETTE INTERFACE

Setting up the Printer

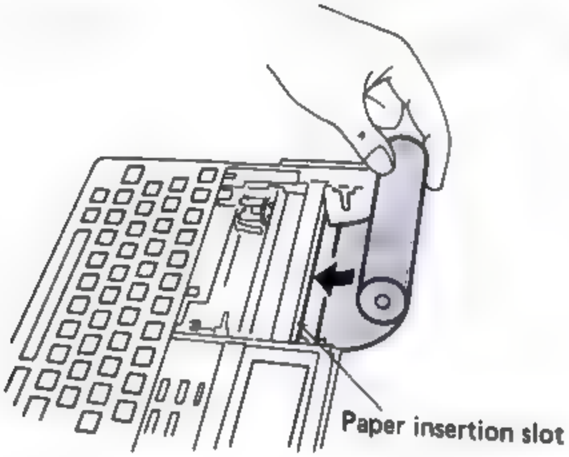
Before using the printer, insert the roll paper and attach the pens as described below.

1. Inserting the Roll Paper

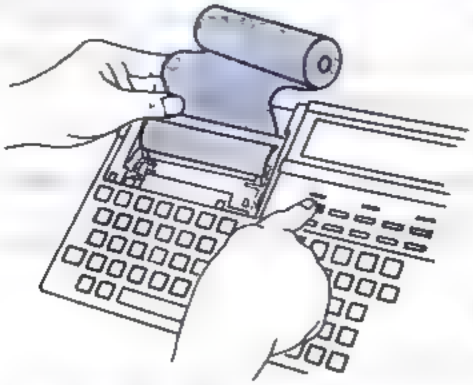
(1) Push the printer cover in the direction of the arrow shown in the figure and remove.



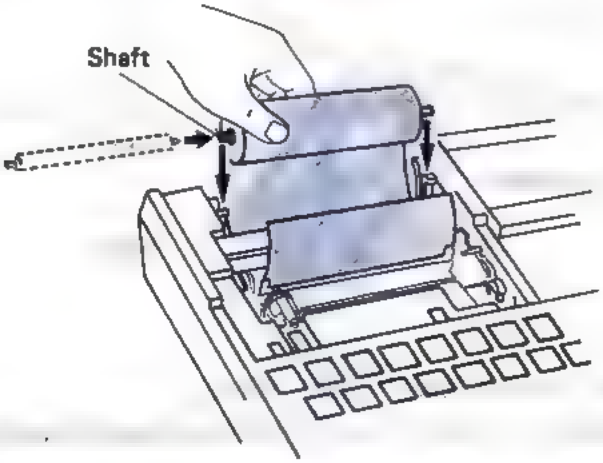
(2) After cutting the beginning of the paper so that it is straight, insert it into the paper insertion slot.
(It may be hard to insert the paper if the beginning is crooked or wrinkled.)



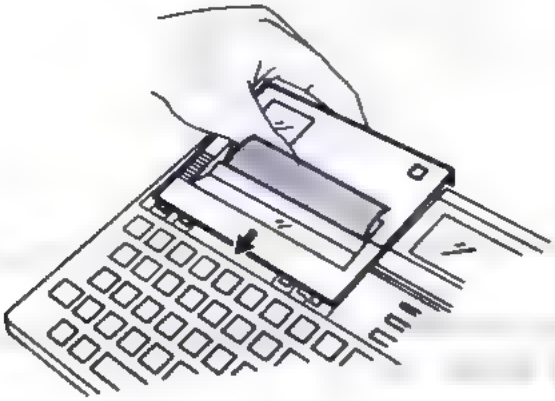
(3) Turn on the power switch on the PC-2500. Press the [illegible] key until the beginning of the paper sticks out about 3 to 5 cm from the printer. (If the paper does not advance, gently help the paper along.)



(4) Insert the shaft into the roll paper and place the roll paper (with shaft) into the paper case.



(5) Attach the printer cover. Make sure the beginning of the paper sticking out of the printer is between the edges of the paper cutter on the printer cover.



* To take out the paper from the printer, pull the paper straight out from the paper case side or from the paper cutter side.

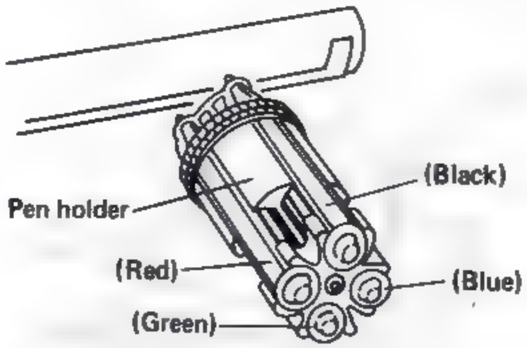
Note: Oil or perspiration (from your hands) on the paper may cause the printing to blur. Wipe off any oil or perspiration. Further, if the printed paper gets wet, it may blur because the ink in the ball-point pen is water based. Be sure to avoid water when handling the paper.

See you nearest dealer for replacement roll paper and specify "Roll paper EA-515P for the PC-2500." (Do not use any other type of paper.)

*EA-515P (3 rolls per set)

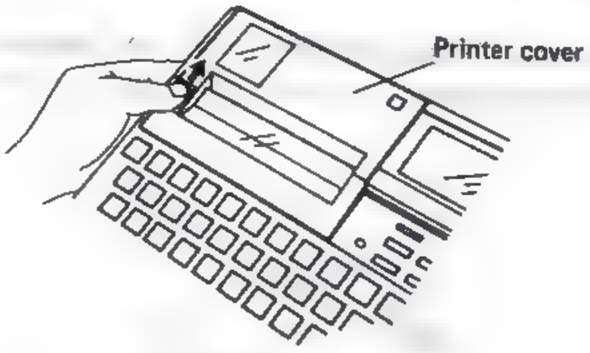
2. Attaching and Replacing Pens

Four ball-point pens can be attached to the built-in printer.
The pen positions on the pen holder are shown in the figure below.



To attach or replace the pens, follow the steps below.

(1) Push the printer cover in the direction of the arrow shown in the figure and remove.



(2) Turn on the power switch on the PC-2500. The following will be displayed:

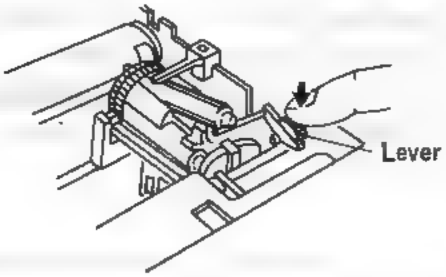
```
    1. BUSINESS SOFTWARE
    2. TELEPHONE BOOK
    3. BASIC
```

* If you are using BASIC or the business software, hold down the SHIFT key and press the ON/BRK key to obtain the above menu.

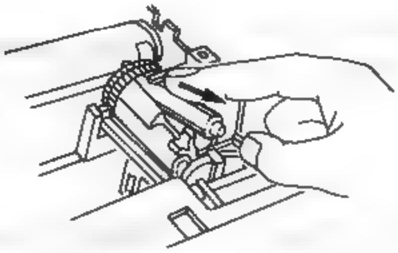
Press the 3 key and enter BASIC.

(3) Press the PEN key while holding down the SHIFT key. The printer changes to the pen replace mode and the pen holder moves to the right margin. (The black pen is at the top.)

(4) Press the pen lever to remove the pen. The pen at the top can be removed.

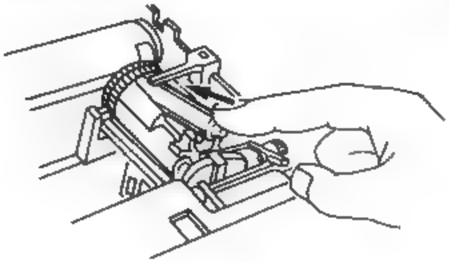


(5) Place your finger on the top of the pen and move it toward you to remove the pen.



(6) Attach a new pen. Place the pen on the holder and push the pen into the pen holder.

* Be sure the pen is of the correct color



(7) To remove or attach the next pen, press the PEN key. The pen holder rotates so that the next pen moves to the top. Repeat steps (4) through (6) to remove and attach the pen.

(8) After you finish replacing the pens, press the PEN key while holding down the SHIFT key. This clears the pen replace mode and the pen holder returns to the left.

Note: If you accidentally rotate the pen holder during pen replacement, the pen position shifts and normal printing operation cannot be obtained. Turn off the power switch and then turn on again.

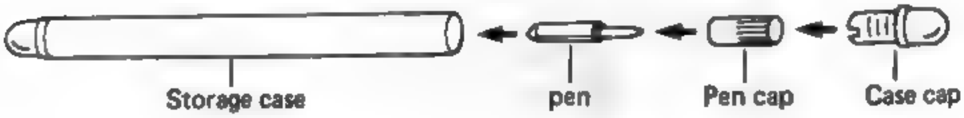
Notes

* Handling the printer

- Do not touch the internal printer mechanism (e.g. drum, pen holder, gear, etc.). Touching the internal mechanism may increase the drawing error and may also cause a shift in the pen position or other printer problems.
- If you happen to drop a pen into the printer either:
  a. Turn the PC-2500 upside down so that the pen falls out or
  b. Gently shake the PC-2500 so that the pen can be retrieved from the hole at the bottom of the unit.

* Handling the pens

Attach the pens when you are going to use the printer. When you are finished with the printer, remove the pens, cap them, and place them in their storage case. If the pens are left in the printer for long periods of time, the ink may dry out.



* The precision of the printer mechanism limits the error in the X direction to 1% and the Y direction to 2%.

* See your nearest dealer for replacement printer pens and specify "Ball-point Pens for the PC-2500." The following pen sets are available:

- EA-850C (black, blue, green, red, 1 each for a total of 4 pens)
- EA-850B (4 black pens)

Printer Related Commands

Four BASIC commands related to the printer are available.

LPRINT: Provides data to the printer. (See page 148.)

LLIST: Prints the program stored within the PC-2500 on the printer. (See page 112.)

TEST: Used to check the colors (color specified by 0–3 in the CHR$&1B command) and check if the ink has run out. If TEST ENTER is entered, 4 squares (5mm x 5mm) will be drawn in different colors. (See page 120.)

□ □ □ □
(1) (2) (3) (4)

(1) color of position 0 (black)
(2) color of position 1 (blue)
(3) color of position 2 (green)
(4) color of position 3 (red)

CONSOLE: Specifies the number of printed characters per line. (See page 194.)
CONSOLE 3Ø: LPRINT
A printing line of 39 characters is specified when entering the BASIC mode.

Note: Use the form
CONSOLE expression: LPRINT
when executing the CONSOLE command for the built-in printer. If executed in this form, the paper is fed by one line. To prevent this line from fecding use the form CONSOLE expression: LPRINT CHR$ 11

The LPRINT command (in a program or manual entry) provides control codes and data to the printer to print characters and numeric values, and draw pictures. Printer control codes and their uses are described next.

(1) Mode Switching

There are 2 modes: one to write characters and numeric values and the other to draw pictures.

Text mode: Mode to write characters, equations, calculation results, etc.
Graphic mode: Mode to draw pictures.

These modes are specified by using the following commands (codes).

Text mode: { CHR$ &1B+"a"
or "A" ↑ 27 (decimal) may also be used

"A": Effective in Graphic mode only
Graphic mode: CHR$ &1B+"b"

Examples: LPRINT "A" Sets the text mode
LPRINT CHR$ &1B+"b" Sets the graphic mode

The text mode is set whenever the power is turned on (except after auto power off).

Note: The "+" in CHR$ &1B+"a" can be rewritten with a semicolon (;). However, various commands may not function properly if the number of displayed (printed) characters has been specified in the USING command.

(2) Effective Commands in the Text Mode

The following printer commands (codes) are effective in the text mode.

① **CHR$ &08 . . . . Backspace (BS)**
The pen backspaces by one character. If the pen is at the left margin, this command is ignored.

② **CHR$ &0A . . . . Line feed (LF)**
The paper is fed one line. The pen is not moved.

③ **CHR$ &0B . . . . Line up (LU)**
The paper is moved down one line. The pen is not moved.

④ **CHR$ &0D . . . . Carriage return (CR)**
The pen is moved to the left margin of the next line.

⑤ **CHR$ &1B . . . . Escape (ESC)**
(This command is also effective in the graphic mode.)
The character(s) following this command sets the mode, character size, color, etc.

- CHR$ &1B+"a" . . . . Sets the text mode.
- CHR$ &1B+"b" . . . . Sets the graphic mode.
- CHR$ &1B+"?"+"a" to "o"

Sets the character size. The character size is set by a letter from "a" to "o" where "b" to "o" specify sizes from 2 to 15 times the size of characters specified by "a" (the smallest size).

Example: LPRINT CHR$ &1B+"?"+"a"

The following are set.
Character size: 1.2 mm (height) x 0.8 mm (width)
Character spacing: 1.2 mm
Line spacing: 2.4 mm

Size "b" is set whenever the power is turned on.

- CHR$ &1B+"0" to "3"

  Sets the color of the characters or drawings.

  ```
  ESC + 0   Black
  ESC + 1   Blue
  ESC + 2   Green
  ESC + 3   Red
  ```

(Example)

```
 10: LPRINT  CHR$ &1B+"a"                       Set the text mode
 20: LPRINT  CHR$ &1B+"?"+"a"                   Set the character size (1x)
 30: LPRINT  CHR$ &1B+"0"                       Set the color (black)
 40: LPRINT  "ABCDEFGHIJKLMNOPQRSTUVWXYZ1       Specify the data (string)
     234567890"
 50: LPRINT  CHR$ &1B+"?"+"b"                   Set the character size (2x)
 60: LPRINT  CHR$ &1B+"2"                       Set the color (green)
 70: LPRINT  "123*456=";                        Specify the data (string)
 80: LPRINT  CHR$ &1B+"3";                      Set the color (red)
 90: LPRINT  123*456                            Specify the data in equation form
100: LPRINT  CHR$ &1B+"?"+"e"                   Set the character size (5x)
110: LPRINT  "END"                              Specify the data (string)
```

Execution of the program above prints the following.

```
ABCDEFGHIJKLMNOPQRSTUVWXYZ1234567890  ← Black

123*456=56088.                        ← Green, Red

E N D                                 ← Red
```

The string in line 70 is printed (in green) after which the multiplication in line 90 is printed (in red).

Character Size and Number of Printed Columns (Characters) Per Line

Line Size	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
Number of Columns Per Line	80	40	26	20	16	13	11	10	8	8	7	6	6	5	5
Height (mm)	1.2	2.4	3.6	4.8	6.0	7.2	8.4	9.6	10.8	12.0	13.2	14.4	15.6	16.8	18.0
Width (mm)	0.8	1.6	2.4	3.2	4.0	4.8	5.6	6.4	7.2	8.0	8.8	9.6	10.4	11.2	12.0

(3) Effective Commands in the Graphic Mode

The graphic mode enables you to draw pictures and graphs on the printer. The following commands, as well as the CHR$ &1B command described above, can be used in this mode.

① **A command: Specifies the TEXT mode**

The plotter is set to the TEXT mode. The same as CHR$ &1B + "a" which is described in the section explaining Control Codes.

Example: LPRINT "A"

Note: Execution of this command in the text mode prints only an "A".

② **D command: absolute movement (pen down)**

This command is used in the following form to draw lines.

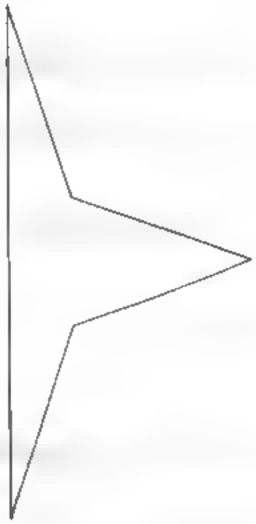
"D X1, Y1, X2, Y2, . . . Xn, Yn"

Xn and Yn represent the X-component and Y-component in an X-Y coordinate system, respectively. This command draws a line from the current pen position to the position specified by coordinates X1, Y1. If X2, Y2 are also specified, the line is drawn in succession to this next position. In this manner, the pen draws lines in succession until the position specified by Xn, Yn.

Example:
LPRINT "D200, 0, 200, 200, 0, 200, 0, 0."
Starting from when the pen is at position 0,0, this command draws a square, 200 steps (40 mm) on a side.

Example:

```
 5:USING
10:CONSOLE 39: LPRINT
20:LPRINT CHR$ &1B+"b"
30:FOR A=1 TO 6
40:READ B,C
50:LPRINT "D";B;",";C
60:NEXT A
70:LPRINT "A"
80:DATA 0,200,50,50,200,0,
   50,-50,0,-200,0,0
```



③ **H command: returns to the origin (home)**

This command lifts up the pen and returns it to the origin.

Example: LPRINT "H"

④ **I command: Defines the origin**

Sets the origin (X = 0, Y = 0) at the current position of the pen. When GRAPH mode is specified by CHR$ &1B+"b," the pen returns to the left margin and this position is set as the origin. When you create a drawing, it may be inconvenient if the original is set at the left edge. In this case, move the pen to any position using an M Instruction (refer to page 78) and use an I command to set that position as the origin. Then you can draw starting with that position.
Example: LPRINT "I"

⑤ J command: relative movement (pen down)

This command is used in the following form to draw lines.

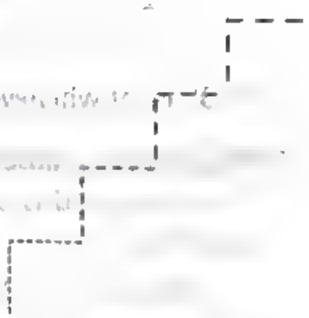
"J X1, Y1, X2, Y2, . . . Xn, Yn"

With the current pen position as the temporary origin, a line is drawn to the position specified by X1, Y1. Next, position X1, Y1 becomes the temporary origin and a line is drawn to X2, Y2 (if specified).
If this manner, the pen draws lines in succession until the position specified by Xn, Yn.

Example:
LPRINT "J200, 0, 0, 200, –200, 0, 0, –200"
Starting from the current pen position, this command draws a square, 200 steps (40 mm) on a side.

Example:

```
10:LPRINT CHR$ &1B+"b"
20:A$="0":B$="45"
30:FOR N=2 TO 8 STEP 2
40:LPRINT "L";N
50:LPRINT "J"+A$+","+B$+","
   +B$+","+A$
60:NEXT N
70:LPRINT "M0,0"
```

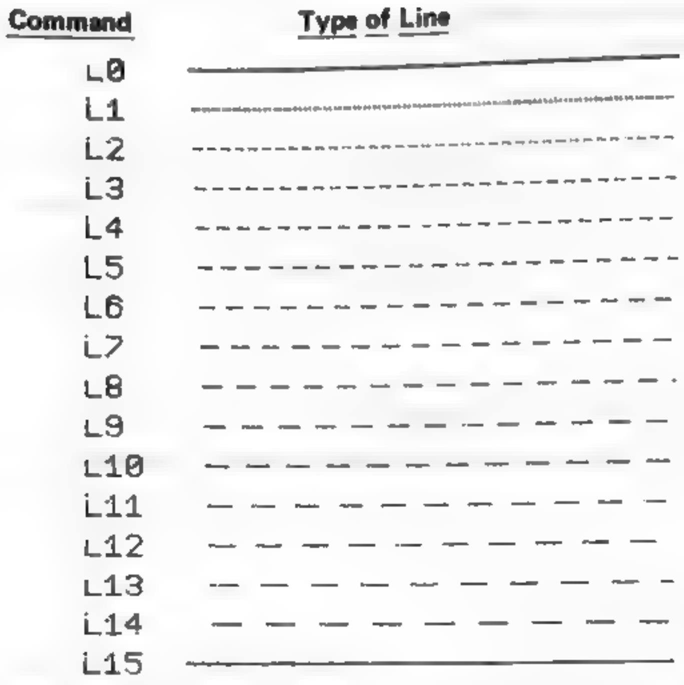


⑥ L command: defines the dotted lines

This command is used in the following form to specify the type of line to be drawn.

"L0" to "L15"

The types of lines available are shown on the next page.



"LØ" is set whenever the power is turned on.

⑦ **M command: absolute movement (pen up)**

This command is used in the following form to move the pen without drawing anything.

"M X1, Y1, X2, Y2, . . . , Xn, Yn"

Example:

LPRINT "MØ, Ø, Ø, 2ØØ"

The pen is first returned to the origin and then moved to position Ø, 2ØØ. (The paper returns 40 mm.)

⑧ **P command: draws letters**

This command is used in the following form to print letters in the graphic mode.

"Pstring" (Same as "P" + "string". "P" + character variable can also be used.)

Example: LPRINT "PABCD" Prints ABCD.

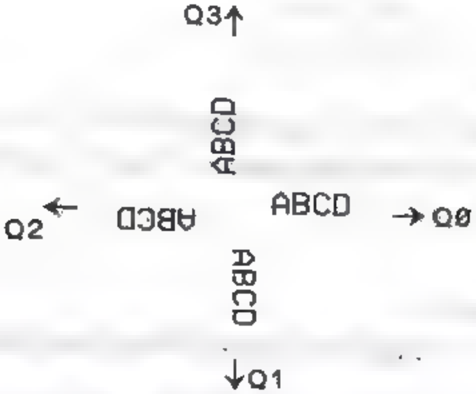
- The character size and color can be specified with the CHR$ &1B command. See page 74 for detailes.
- When the characters exceed the number printable in one line, "scissoring" occurs. (See page 81.)

⑨ **Q command: rotate (rotates characters)**

This command is used in the following form to specify the orientation (printing direction) of characters.

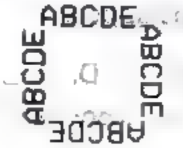
"QØ" to "Q3" (same as "Q"+"Ø" to "3")

Characters are printed in one of the directions shown below (by arrow) depending on the command specified (from QØ to Q3).



(Example)

```
10:LPRINT CHR$ &1B+"b"
20:AB$="ABCDE"
30:LPRINT "M120,0"
40:FOR A=0 TO 3
50:N$= STR$ A
60:LPRINT "Q"+N$
70:LPRINT CHR$ &1B+N$
80:LPRINT "P"+AB$
90:NEXT A
```



⑩ **R comand: relative movement (pen up)**

This command is used in the following form to move the pen without writing anything.

"R X1, Y1, X2, Y2, ..., Xn, Yn"

With the current pen position as the temporary origin, the pen is moved to the position specified by X1, Y1. Next, position X1, Y1 becomes the temporary origin and the pen is moved to X2, Y2 (if specified).

Note that even if you make a mistake when writing a command letter or data (insufficient data, missing comma, extra character, etc.) and execute the command, no error will be indicated.
However, if you attempt to move the pen (or paper) more than +409 mm from the origin in the graphic mode, the system will reset (as if the power switch has been powered on).

(4) Others

(1) The following commands are effective both in the TEXT mode and in the GRAPH mode.

CHR$ &1B+"?"+"a" to "o"	(character scale setting)
CHR$ &1B+"0" to "3"	(color setting)
CHR$ &1B+"a"	(TEXT mode)
CHR$ &1B+"b"	(GRAPH mode)

When the command CHR$ &1B+"b" is performed, "Q0" is set. The pen returns to left and this position is set as the origin. The L command remains effective.

(2) Data reading
- Leading blanks (" ") are ignored.
- If a number has more than four digits, the lower of the three digits are effective.
- A comma (,) or CR code is used of the end at each data. When characters other than numbers are read, all such values are ignored until the next comma or CR code appears.

  Example: D␣␣␣ – 135. 21, 20, ...

  Ignored

(3) TEXT/GRAPH setting commands are effective only after a CR (Carriage Return) or LF (Line Feed) code.

(4) During Program execution in the GRAPHIC mode, the [illegible] key may not function after the execution is forcibly stopped (e.g. with the [illegible] key).
Either key in LPRINT [ENTER] or turn the power switch off and then on.

(5) The values that can be specified in M, D, R or J command are between –999 and +999. But if you input more than one R or J command, values between –2048 and +2047 can be specified.
If a value exceeding these ranges is specified, the printer resets (as if the reset button has been pressed or the power switch has been turned off and then on).

(6) The following settings change after the mode is changed from the GRAPHIC mode to the TEXT mode.
- The X and Y coordinates in the GRAPHIC mode change to 0, 0.
- The printing direction changes to Q0.
- When the TEXT mode set command ends with a semicolon (;), the following LPRINT command starts printing from the current pen position.

Example:

```
10 LPRINT CHR$ Q1B+"a";
20 LPRINT "ABC"
```

- The type of line, color, and character size do not change.

(7) Auto-carriage return

The printer automatically returns the carriage in the TEXT mode after the very last column in a line is printed.

For example if the number of printing columns per line is specified in the CONSOLE command at 4Ø columns, the paper will be fed by one line for a line exceeding 4Ø columns (e.g. when executing the LLIST command using standard character size "b" for 4Ø columns per line).

Note: The GRAPHIC mode is for drawing pictures and not for printing results of manual calculations.

If the print switch is set to "P" in the GRAPHIC mode and manual calculations are made, the printer will either print nothing or operate erratically.

Therefore, set the print switch to "•" in the GRAPHIC mode. To print results of manual calculations, use the following command to change to the TEXT mode and then print.

```
LPRINT CHR$ &1B+"a"
```

If the program is stopped with the BREAK key, use the following command to change to the TEXT mode.

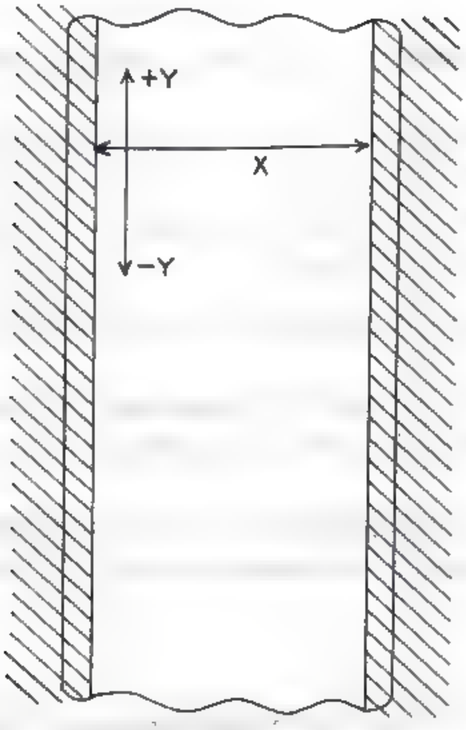
```
LPRINT CHR$ &ØD+CHR$ &1B+"a"
```

- It is convenient to "reserve" the following command to initialize the mode, pen position, color, and character size to the TEXT mode, pen at the left margin, black, and character size "b", respectively (e.g. after ending the program). (See page 65 for information on reserve.)

```
LPRINT CHR$ &1B+"a" ; CHR$ &1B+"Ø" ; CHR$ &1B+"?b" @
```

Range of Movements for the Pen and Paper

Pictures drawn with the printer are limited by the width of the paper used, or approx. 114 mm. However, sometimes it's easier to write a program to draw an entire picture even though we want only part of the picture and other times we want to divide the picture into strips, the width of the paper, so that we can draw a large picture. We can use a convenient function called the scissoring function. This function actually draws the part of the picture which falls on the paper and hypothetically moves the pen and draws the part of the picture which falls beyond the edges of the paper.



(Printing Area)
X direction: 480 steps
Y direction: 999 steps
(1 step = 0.2 mm)

(Shaded areas represent the "scissoring area.")

- Although the scissoring function is convenient, a program error will result in the picture being drawn completely in the imaginary area and not on paper. Exercise care when writing programs.

Minimizing repeated movements in the Y direction (forward and reverse paper feeds) when drawing pictures with the built-in printer will result in more accurate drawings. Although the printer mechanism is highly accurate, approx. 1% error in the X direction and approx. 2% in the Y direction are generated when drawing pictures. Therefore, try to minimize repeated back and forth movements in the Y direction in your programs.

Serial I/O Function

The PC-2500 is equipped with a serial I/O interface. Use of this interface enables you to draw pictures on the optional CE-515P color plotter printer.
You can also use it to connect the PC-2500 to other personal computers to transfer data.

Note: The voltage levels of the signals in this interface and those in the RS-232C interface are different. An adapter is required to convert the voltage when connecting the PC-2500 to an RS-232C interface. A connection cable is also required.

Note: If a voltage exceeding the allowable range of the PC-2500 is applied to its I/O interface connector, the internal parts of the PC-2500 may become damaged.

Basics on the Use of the Serial I/O Interface

Usually the circuit for the serial I/O interface is closed. In this state, the serial I/O interface cannot send or receive data.
Therefore, it is first necessary to open the circuit. Execute the OPEN command to open the circuit. (ERROR 8 results if the circuit is already open.)
The conditions for the I/O operations must match with those of the connected equipment. If the conditions differ between the PC-2500 and the connected equipment, the signals (data) may not be read or errors may result.
The I/O conditions can be set or changed with the OPEN command. I/O operations for data and programs can be performed after you set the conditions for both the PC-2500 and the connected equipment, then open the circuit. The following commands can then be executed.

```
LPRINT, LLIST, SAVE, LOAD,
PRINT #1, INPUT #1
```

After the I/O operations for the data or program are completed, the serial I/O interface circuit is closed.
Although the CLOSE command is executed to close the circuit, the circuit also closes after the program ends (such as when the END command is executed) or after the RUN command is executed.
When writing a program which uses the serial I/O interface, the circuit must be opened, the I/O operation performed, and the circuit must be closed, as described above.

Examples:
The program below inputs characters from the PC-2500 and prints them on the CE-515P. Up to 72 characters (alphabetic and numbers) or 36 special characters can be input at a time. Up to 48 columns are printed in a line..

Note: See page 85 for details on connecting the CE-515P and on setting the conditions.

```
 1Ø OPEN "12ØØ, N, 8, 1, A, C"      ← Open the serial I/O interface circuit and
                                      set the I/O conditions.
 2Ø CONSOLE 48                      ← Set the printing line to 48 columns.
 3Ø CLS : CLEAR                     ← Clear the display and variables.
 4Ø DIM Z$ (Ø) * 72                 ← Allocate variables (number of characters
                                      becomes 72)
 5Ø INPUT "PRINT DATA?", Z$ (Ø)     ← Input data
 6Ø IF Z$ (Ø) = "..." THEN 9Ø       ← When the data is "..." go to line 90.
 7Ø LPRINT Z$ (Ø)                   ← Send the input data.
 8Ø GOTO 3Ø                         ← Return to line 30.
 9Ø INPUT "END   ? (Y/N)"; A$       ← Confirm whether to end the program.
1ØØ IF A$< > "Y" THEN 3Ø            ← If Y was input for line 90, go to line 110.
                                      If N (or anything other than Y) was
                                      input, to line 30.
11Ø CLOSE                           ← Close the serial I/O interface circuit.
```

The program below specifies the test mode, character size, and color for the CE-515P.

Note: To write this program after the program above, change line 110 CLOSE to 110 END.

```
2ØØ "Z" : CLOSE                     ← See note below.
21Ø OPEN "12ØØ, N, 8, 1, A, C"      ← Open the circuit and set the conditions.
22Ø LPRINT CHR$ &1B ; "a"           ← Specify the text mode
23Ø INPUT "CHARACTER SIZE?
    (a..o)" ; B$
24Ø IF ("a"> B$) OR (B$> "o")
    THEN 23Ø
25Ø LPRINT CHR$ &1B; "?" +B$        ← Specify the character size.
26Ø INPUT "COLOR? (Ø..3)" ; C$
27Ø IF ("Ø"> C$) OR (C$> "3")
    THEN 26Ø
28Ø LPRINT CHR$ &1B ; C$            ← Specify the color
29Ø END                             ← End of program (close the circuit)
```

Note: When execution of this program is started by pressing DEF Z while the program execution is temporarily stopped, the open command in line 210 may cause an error since the circuit is closed. The CLOSE command in line 200 is used to close the circuit before execution of the OPEN command.

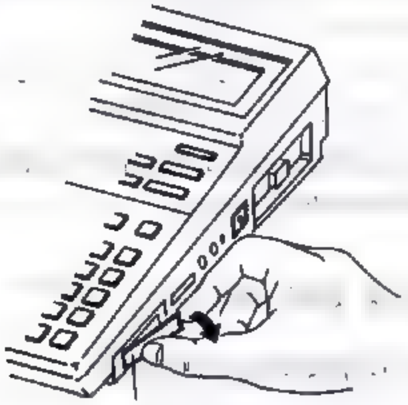
Note: The PC-2500 is not equipped with a timer to time the I/O commands to the serial I/O interface and interrupt communication with the connected equipment. Therefore, if the connected equipment is not ready for or has interrupted communication (e.g. power not turned on, etc.) when an I/O command is executed, stop the execution by pressing the ON/BRK key.

Connecting the CE-515P and Setting the Conditions

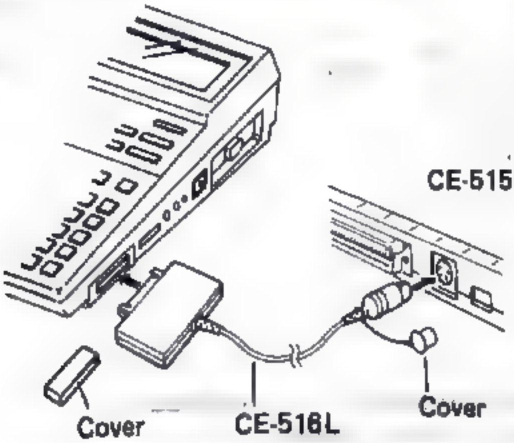
The optional connection cable, CE-516L, is used to connect the PC-2500 and the CE-515P.

First, turn off the power of both the PC-2500 and the CE-515P.

Next, remove the cover from the serial I/O interface connector.



Then, remove the cover from the CE-515P. Connect the serial I/O interface connector to the PC-2500 and the RS-232C connector to the CE-515P.

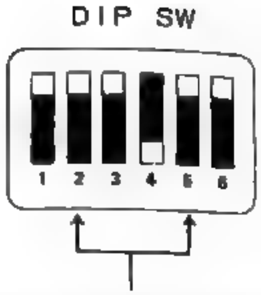


Next, set the 3rd and 4th DIP switches on the rear of the CE-515P to their lower positions. Set the rest of the switches to their upper positions. (See figure on next page.)

(Be sure that at least the 2nd and 5th switches are at their upper positions.)

- Once you set the DIP Switches as shown on the left, the following input conditions will be set.

1: 8-bit data length (word length)
2: RS-232C interface selection.
3: ASCII code
4: Feed a line and return the pen to the beginning of the line after receiving the CR code
5: TTL level input signals
6: Not used



Be sure to set these switches (2nd and 5th) to their upper positions.

Turn on the CE-515P, attach the pen, and position the printer paper.

Note: Character Codes
Some character codes differ between the PC-2500 and the CE-515P.
As a result, the displayed character (on the PC-2500) and printed character (on the CE-515P) may differ when the CHR$ command is used. The different character codes are shown in the table below.

Character Code	Printer on the PC-2500	CE-515P
&Ø1 ~ &1F	The following codes are control codes. The others are nulls. &Ø8, &ØA, &ØB, &ØD, &1B Shown as spaces on the display.	The following codes are control codes. The others are nulls. &Ø8, &ØA, &ØB, &ØD, &ØE, &ØF, &1B
&6Ø	" ` "	" ` "
&7F	Space	☒
&8Ø ~ &9F	Space	null
&EØ ~ &FF	&E0 ~ &F4 } Space &F0 ~ &FF } &FB ⟶ π &FC ⟶ √‾ &F5 ⟶ ♤ &F6 ⟶ ♡ &F7 ⟶ ◇ &F8 ⟶ ♧ NOTE: Codes &F9 and &FA are printed as spaces on the PC-2500.	null

Specify character codes shown in the character code table for the CE-515P when printing on the CE-515P.

- The CE-515P does not have characters √‾and π. Thus, when printing programs using these characters, they are coverted to SQR and PI, respectively, and then printed.

Using the Cassette Interface

Using this cassette interface will allow you to store programs and data from the computer onto cassette tape (of course you'll also need a cassette recorder such as we sell for this portable computer system: model CE-152). Once on tape, you can load these programs and data back into the computer with a simple procedure.

Connecting the PC-2500 to a Tape Recorder

Only three connections are necessary:

1. Connect red plug into the MICrophone jack on the cassette recorder.
2. Connect gray plug into the EARphone jack on the cassette recorder.
3. Connect black plug into the REMote jack on the cassette recorder.

Cassette Tape Recorder

We recommend that you use the optional cassette tape recorder CE-152 for your portable computer system. The CE-152 designed to match the PC-2500 records programs and data. Any recorded program can be retrieved and reloaded into the PC-2500.

When you use any cassette tape recorder other than the CE-152:

The following is a description of the required minimum tape recorder specifications.

Item	Requirements
1. Recorder Type	Any tape recorder, standard cassette or microcassette recorder, may be used in accordance with the requirements outlined below.
2. Input Jack	The recorder should have a mini-jack input labeled "MIC". Never use the "AUX" jack.
3. Input Impedance	The input jack should be a low impedance input (200 ~ 1,000 OHM.)
4. Minimum Input Level	Below 3 mV or −50 dB.
5. Output jack	Should be a minijack labeled "EXT. (EXTernal speaker)", "MONITOR", "EAR (EARphone)" or equivalent.
6. Output impedance	Should be below 10 OHM.
7. Output level	Should be above 1V (practical maximum output above 100 mW)
8. Distortion	Should be within 15% within a range of 2 KHz through 4 kHz.
9. Wow and Flutter	0.3% maximum (W.R.M.S.)
10. Other	Recorder motor should not fluctuate in speed.

- In case the miniplug provided with the PC-2500 is not compatible with the input/output jacks of your tape recorder, special line conversion plugs are available on the market.

Note: Some tape recorders may not perform properly due to different specifications. Those tape recorders having distortion, increased noise, and power deterioration after long years of use may not show satisfactory results owing to changes in their electrical characteristics.

Operating the Cassette Interface and Recorder

Recording (saving) onto magnetic tape

See Tape Notes.

1. Turn off the REMOTE switch on the PC-2500.
2. Enter a program or data into the Computer.
3. Load tape into the tape recorder.
   Determine the position on the tape where you want to record the program.
   - When using a tape, be sure the tape moves past the clear leader (non-magnetic mylar material).
   - When using a tape already partially recorded, search for a location where no recording exists.
4. Connect the Interface's red plug to the tape recorder's MIC jack and the black plug to the REM jack.
5. Turn on the REMOTE switch.
6. Simultaneously press record and play buttons on the tape recorder (to put it in record mode).
7. Enter recording instructions (CSAVE statement, PRINT# statement), and press the [ENTER] key for execution.

   First set the unit to "RUN" or "PRO" mode. Next push the following keys:
   [C] [S] [A] [V] [E] [SHIFT] ["] file name [SHIFT] ["] [ENTER] .

   (To write the contents of data memory onto tape, push as follows:
   [P] [R] [I] [N] [T] [SHIFT] [#] [ENTER] .)

   E.g., [C] [S] [A] [V] [E] [SHIFT] ["] [A] [A] [SHIFT] ["] [ENTER]

   When you press the [ENTER] key, tape motion will begin, leaving about a 8-second non-signal blank. (Beep tone is recorded.) After that, the file name and its contents are recorded.
8. When the recording is complete, the PROMPT symbol (>) will be displayed and the tape recorder will automatically stop. Now you have your program on tape (it still is in the Portable Computer also).
   When data are to be automatically recorded by program execution (PRINT # statement, not manual operation), set up steps 1 thru 6 before executing the program.

   To aid you in locating programs on tapes, use the tape counter on the recorder.

Collating the Computer and Tape Contents

See tape Notes.

After loading or transferring a program to or from tape, you can verify that the program on tape and the program in the Portable Computer are identical (and thus be sure that everything is OK before continuing your programming or execution of programs).

1. Turn off the REMOTE switch.
2. With cassette in the recorder, operate the tape motion controls to position tape at the point just before the appropriate file name to be checked.
3. Connect gray plug to EARphone and black plug to REMote jacks.
4. Turn on the REMOTE switch.
5. Press PLAY button of recorder.
6. Input a CLOAD? statement and start execution with ENTER key. Do this as follows: Set unit to "RUN" or "PRO" mode. Enter the following key sequence—

   The file name which you used previously.

   C L O A D SHIFT ? SHIFT " A A SHIFT " ENTER

   The Pocket Computer will automatically search for the specified file name and will compare the contents on tape with the contents in memory.

   During collation, the mark "*" is shown at the rightmost digit of the bottom line of the display. The mark "*" will disappear when collation is completed. While a file name is being retrieved, no "*" mark will be displayed as collation has not started yet.
   (The same occurs when the first program is read without a file name.)

   If the programs are verified as being identical, a PROMPT symbol (>) will be displayed on the Portable Computer.

   If the programs differ, execution will be interrupted and an Error code 8 will be displayed. If this occurs, try again.

Loading from a magnetic tape

See Tape Notes.

To load, transfer, or read out programs and data from magnetic tape into the Portable Computer, use the following procedure.

1. Turn off the REMOTE switch.
2. Load tape in the tape recorder. Position tape just before the portion to be read out.

3. Connect the gray plug to the EAR jack on the tape recorder, and the black plug to the REM jack.
   [In using a tape recorder having no REM terminal, press the PAUSE button to make a temporary stop.]
4. Turn on the REMOTE switch.
5. Push the PLAY button on the tape recorder (to put unit in playback mode).

   Set the VOLUME control to middle or maximum.

   Set Tone to maximum treble.
6. Input transfer instructions (CLOAD statement, INPUT # statement), and press ENTER key for execution.

   Put the unit into "RUN" or "PRO" mode. Then push the following keys:
   C L O A D SHIFT " file name SHIFT " ENTER .

   (To load the contents of the data memory, push as follows: I N P U T SHIFT # ENTER .)

   E.g., C L O A D SHIFT " A A SHIFT " ENTER

   The specified file name will be automatically searched for and its contents will be transferred into the Portable Computer.
   The mark "*" appears while loading the designated CSAVEd program from the tape to the computer's memory.
   (The same occurs when the first program is read without a file name,) The mark "*" disappears when the load is performed completely.
7. When the program has been transfered the computer will automatically stop the tape motion and display the PROMPT (>) symbol.

   To transfer data (INPUT # statement) in the course of a program, set up steps 1 thru 5 prior to executing the program.

Notes:
- If an error occurs (error code "8" is displayed), start over from the beginning. If the error continues, adjust volume up or down slightly.
- If the error code is not displayed but tape motion continues the transfer is not occuring correctly. Press ON key (to "break") to stop the tape. Repeat steps.
- If the error remains or the tape continues to run after several attempts to correct the problem, try cleaning and demagnetizing the Recorder's tape head.

Tape Notes

1) For any transfer or collation, use the tape recorder that was used for recording. If the tape recorder for transfer or collation is different from that used for recording, transfer or collation may not be possible.

2) Always use only the highest quality tape for programs and data storage (economy grade audio type tape may not provide the proper characteristics for digital recordings).

3) Keep the tape heads and tape handling parts clean—use a cassette cleaner tape to keep everything clean.

4) Volume setting—set to middle or maximum level.
Volume level can be very important when reading in data from the recorder; make slight adjustments as required to obtain error-free data transfer. A slight adjustment either up or down may result in perfect recordings every time.

5) Be sure all connections between the pocket computer and cassette interface are secure. And be sure the connections between interface and recorder are secure and dirt-free.

6) If problems occur when using AC power for the CE-126P and/or the recorder, use battery power instead (sometimes the AC power connection also adds some "hum" to the signal which upsets proper digital recordings).

- To connect the AC adaptor to the CE-126P, turn the CE-126P power off and then connect the adaptor to the CE-126P.

7) Tone control—set to maximum treble.

8) When recording programs or data on a used tape, erase the portion before writing and execute the recording command. (Make sure that the previous program is completely erased without any portion remaining.)

CHAPTER 8 USING THE RAM CARD (OPTION)

The PC-2500 is capable of using optional RAM cards CE-201M or CE-202M.

> The CE-201M and CE-202M are RAM (Random Access Memory) cards with capacities of 8 K bytes and 16 K bytes, respectively.

The program/data area of the PC-2500 can be expanded by mounting the RAM card. Further, a program can be saved even if the RAM card is removed after loading the program so that different programs can be executed simply by changing the RAM card.
Table data created by the business software can also be stored by using a RAM card.

Note: Programs and data within the RAM card can be backed up by the built-in batteries.

- Be sure to read the RAM card manual for information on its use.

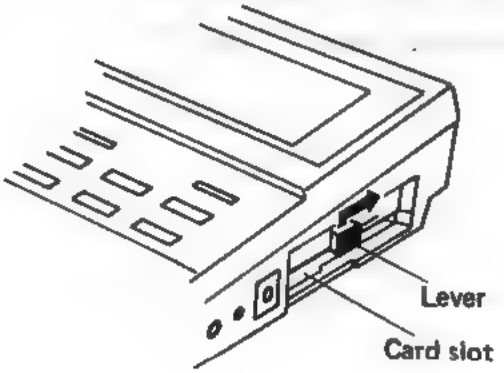
Note: Be sure the power switch is off before inserting or removing the RAM card.

1. Mounting the RAM Card

If the RAM card is to be used for the first time, be sure to insert the battery into the RAM card.

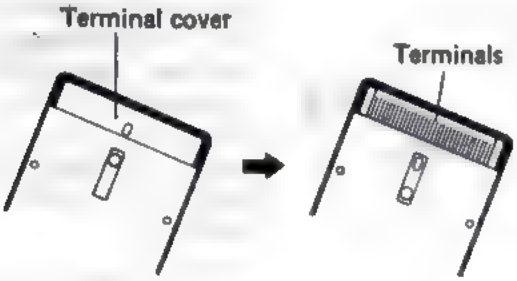
(1) Turn off the PC-2500 before installing or removing the RAM card.

(2) Move the lever at the card slot in the direction of the arrow shown in the figure.

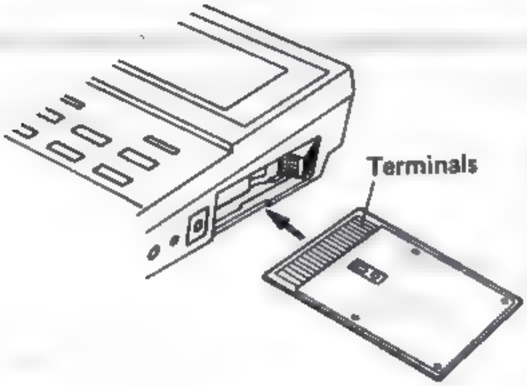


(3) Open the Terminal cover of the RAM card.

Note: Completely open the terminal cover (so that it does not cover the terminals). Also, do not touch the terminals.

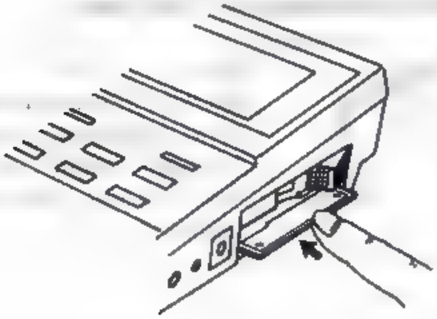


(4) Insert the RAM card, with its terminals facing up, into the card slot.

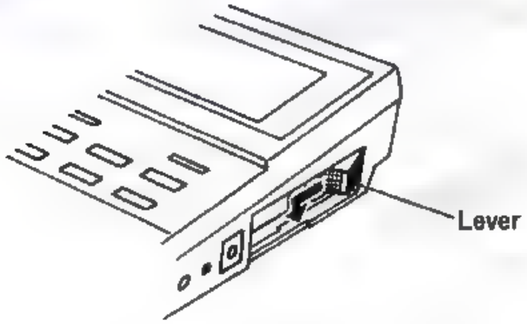


IMPORTANT:
Do not insert the RAM card backward nor insert it with the terminal cover not completely open. Doing so may short the terminals and damage the PC-2500.

Firmly push the RAM card in to install it properly.



(5) Move the lever back to its original position.

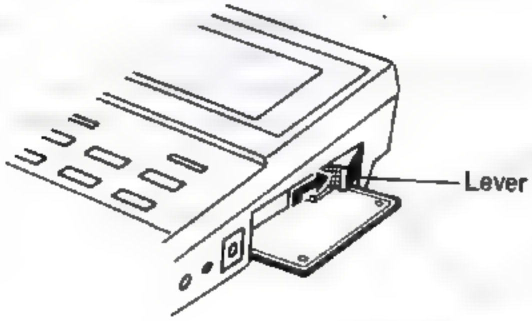


Note:
Be sure to move the lever back to its original position. If you do not, the PC-2500 will not function. If the PC-2500 is turned on without moving back the lever, move back the lever, and turn the power switch off and then on.

2. Removing the RAM Card

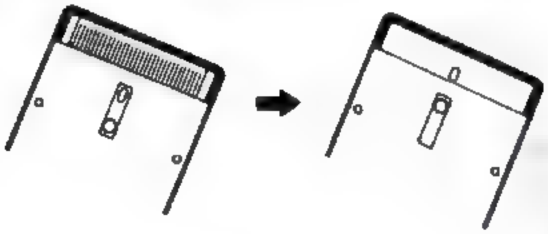
(1) Turn the PC-2500 off.

(2) Move the lever at the card slot in the direction of the arrow shown in the figure.



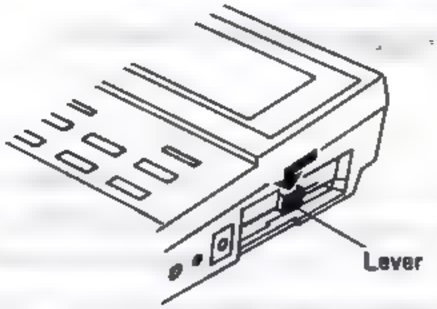
Remove the RAM card.

(3) Immediately close the terminal cover after removing the RAM card.



Note:
Be sure to close the terminal cover completely (so that the terminals cannot be seen). Also, do not touch the terminals.

(4) Move the lever back to its original position.

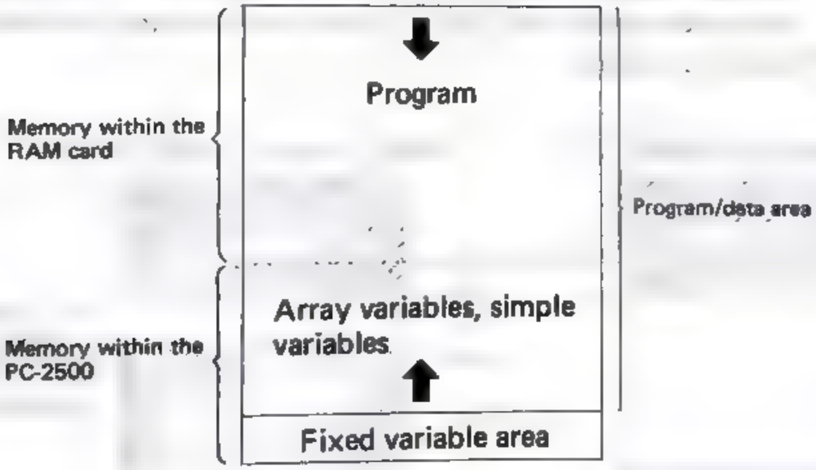


Note:
Be sure to move the lever back to its original position. If you do not, the PC-2500 will not function. If the PC-2500 is turned on without moving back the lever, move back the lever, and turn the power switch off and then on.

3. Using the RAM Card

Normally, the memory in the PC-2500 and the memory in the installed RAM card are linked when the RAM card is mounted in the PC-2500.

As shown in the figure below the program is written on the memory within the RAM card while array variables and simple variables are allocated in the memory within the PC-2500.


Memory within the RAM card
Program
Program/data area
Memory within the PC-2500
Array variables, simple variables
Fixed variable area

The program written within the RAM card is retained (by the battery inserted into the RAM card) even if the RAM card is removed from the PC-2500. Therefore, the same program can be used by mounting the RAM card again into the PC-2500. In this case, array variables and simple variables are not retained. The table data created by the business program is also not retained.

If a program which exceeds the capacity of the RAM card is written, it will be destroyed and cannot be used when the RAM card is removed.

Keep the size of the program within the capacity of the RAM card to preserve it even after the RAM card is removed. (See page 98.)

Note: Since the reserve area is within the PC-2500, its contents cannot be stored in the RAM card.

- See page 98 for the capacity of the program/data area (in bytes).

4. Saving Data in the RAM card

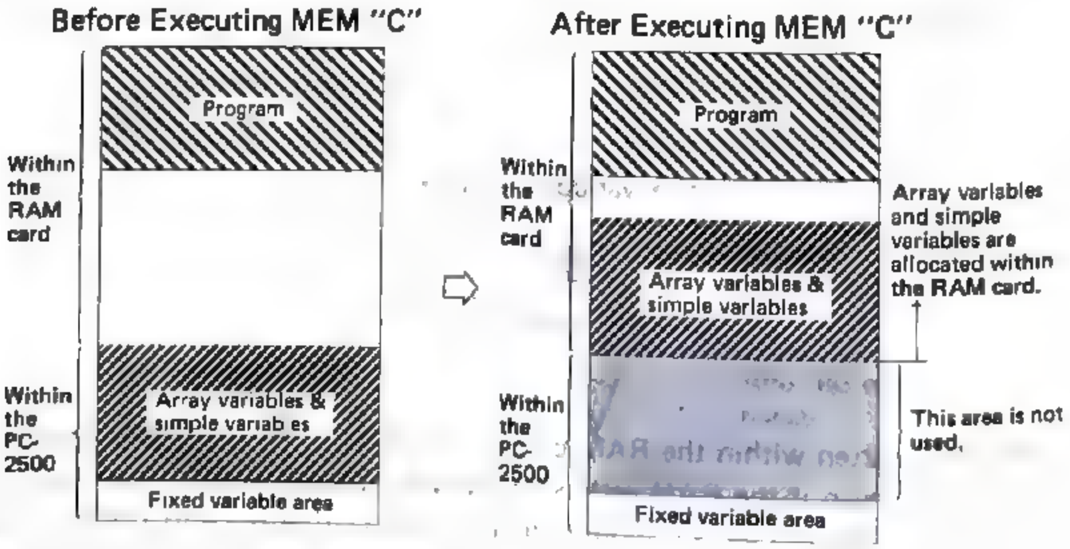
As shown in the previous figure, variables (such as array variables and simple variables) are usually allocated in the memory within the PC-2500. Therefore, if the RAM card is removed from the PC-2500, the variables will be cleared. Execute the following command whenever you want to store data within the RAM card.

```
MEM "C" ENTER
```

Executing the MEM "C" command allocates the array variables and simple variables in the following way.

- The array variables and simple variables allocated within the PC-2500 are transferred to the RAM card.
- The array variables and simple variables are all allocated within the RAM card. (The variables are not allocated within the PC-2500.)

Therefore, the variables (data) are retained even if the RAM card is removed from the PC-2500. Table data created with the business program can also be allocated within the RAM card and retained.



The settings by the MEM "C" command are cleared by executing the MEM "B" command. The following are performed.

- The array variables and simple variables allocated within the RAM card are transferred to the PC-2500.

- The array variables and simple variables are allocated within the PC-2500.

The setting for the MEM "C" command is stored within the RAM card. Removing the RAM card from the PC-2500 invalidates the setting. Installing the RAM card once again restores the setting.

- To keep the data stored within the RAM card, define a key and start the program using the defined key. When the program is executed using the RUN command, the array variables and simple variables will be erased. Therefore, do not use the RUN command.

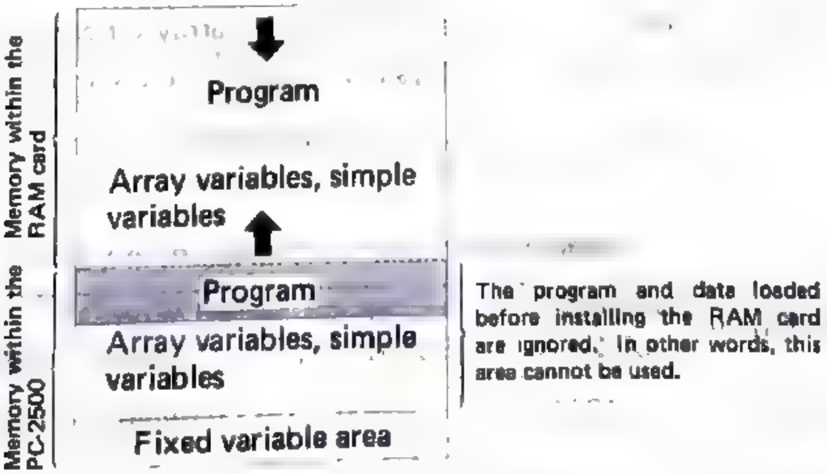
- Fixed variables cannot be saved on the RAM card.
  Therefore, to save data, use array variables or simple variables.

If the array variables, simple variables, and table data for the business software do not fit into the RAM card when MEM "C" is executed, an error results (ERROR 6) and the system returns to the state before execution.

5. Precautions When Using the RAM Card

- If you want to install the RAM card as well as retain the program within the PC-2500,
  ① Save the program (within the PC-2500) on tape, or
  ② Install the RAM card with the setting by the MEM "C" command.

  If the memory within the PC-2500 contains programs and data when the RAM card is installed, they will be ignored while another program is input and executed. (See figure below.)


Memory within the RAM card
Program
Array variables, simple variables
Memory within the PC-2500
Program
Array variables, simple variables
Fixed variable area
The program and data loaded before installing the RAM card are ignored. In other words, this area cannot be used.

Array variables and simple variables (table data created by the business software and the program loaded in the RAM card are retained.) in the PC-2500 will be cleared when the RAM card with the setting by the MEM "B" command is installed. Thus, if the contents in the PC-2500 are important, save them on tape before installing the RAM card.

- The stored program and data are cleared when the battery in the RAM card is replaced (or removed). Therefore, if a valuable program is stored in the RAM card, it is recommended that the program be recorded beforehand on tape.
  If the battery in the RAM card is replaced while the RAM card is installed in the PC-2500, the contents are not cleared.
- When the new battery is replaced, the contents of the RAM card can be retained for approx. 34 months for the CE-201M and 18 months for the CE-202M (when removed from the PC-2500).

If it becomes necessary to press the all reset switch with the RAM card installed, be sure to press it while holding down only the space bar. If only the all reset switch is pressed, or only the SHIFT , [illegible] keys are pressed, or several keys are pressed and then the all reset switch, the program and data stored within the RAM card are cleared.

Reference

(1) The program size (in bytes) can be obtained by the following operation.

Example: BASIC only

RUN mode

CLEAR ENTER (Clears the simple variables, array variables, etc.)

31Ø2–MEM ENTER ← displays the number of bytes in the program

↑ This value varies depending on the use of the RAM card. Use the values (bytes) below.

Capacity of Program/Data Area (bytes)

PC-2500 only	PC-2500 + CE-201M	within CE-201M only	PC-2500 + CE-202M	within CE-202M only
31Ø2	11294	8142*	19486	16334*

* Capacities within the RAM card when MEM "C" is set.

(2) Using Both BASIC and the Business software

RUN mode

CLEAR [ENTER]	(Clears the simple variables and array variables.)
[SHIFT] [illegible] [1]	(Enters the Business software.)
[DEF] [illegible]	(Specifies the special function.)
[1]	(Checks the remaining memory.)

```
|BASIC          : OOOOO*|
```

*: The value shown here represents the size of the program (in bytes).

If the program size does not exceed the values below, the program will be retained within the RAM card even after it is removed from the PC-2500.

When using CE-201M: 8142 bytes
When using CE-202M: 16334 bytes

If your program is larger than these values, it cannot be stored within the RAM card. Shorten your program or save it on tape.

Set MEM "C" to store

- Data processed by the BASIC program
- Table data created by the Business software

within the RAM card.
If the RAM card is removed without setting MEM "C", the data will be destroyed.

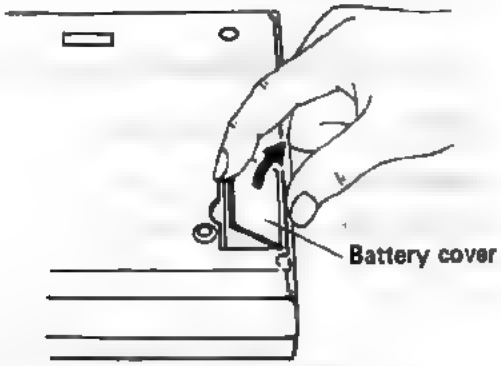
Note: The telephone book will be allocated within the program/data area once you enter the Business software. If you do not use the Business software, follow the method on page 353 to clear the table data and then return to BASIC. This maximizes the capacity of the program/data area that can be used by BASIC (values given in the previous table).

Replacing the Battery in the RAM Card While it is Mounted in the PC-2500

Follow the steps below to replace the battery in the RAM card to retain the stored program and data.

(1) Mount the RAM card into the PC-2500.

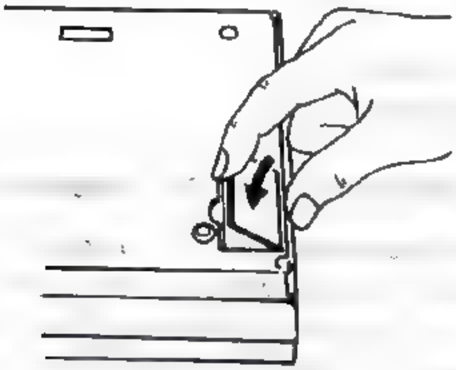
(2) Remove the battery cover (for the RAM card) located on the rear of the PC-2500.



The RAM card's battery cover comes out.

(3) Replace the battery according to the instructions in the RAM card manual.

(4) Attach the battery cover.



Reference

The PC-2500 resets the following when the menu screen (for the Business software and BASIC selection) is displayed.

1. USING
2. PRINT=LPRINT (becomes PRINT=PRINT)
3. WAIT
4. Last answer (resets to Ø)
5. GRAPHIC mode of the printer (sets the text mode)
6. Console value (sets to 39 columns)

CHAPTER 9
BASIC REFERENCE

The following chapter is divided into three sections:

Commands: Instructions which are used outside a program to change the working environment, perform utilities, or control programs.

Verbs: Action words used in programs to construction BASIC statements.

Functions: Special operators used in BASIC programs to change one variable into another.

However, commands related to graphic, serial I/O, and text functions are summarized in the following pages.

Graphic	pages 181 ~ 192
Serial I/O	pages 193 ~ 207
Text function	pages 208 ~ 210

Commands and verbs are arranged alphabetically. Each entry is on a separate page for easy reference. The contents of each section are shown in the tables below so that you can quickly identify the category to which an operator belongs. Functions are grouped according to four categories and arranged alphabetically within each category.

Commands

Program Control

CONT
GOTO*
NEW
RUN

Cassette Control

CLOAD
CLOAD?
CSAVE
INPUT #*
MERGE
PRINT #*

Debugging

LIST
LLIST
TROFF*
TRON*

Graphic

GCURSOR*
GPRINT*
LINE*
POINT
PRESET*
PSET*

Serial I/O

CLOSE*
CONSOLE*
INPUT #1*
LLIST
LOAD
LPRINT*
OPEN*
OPEN$
PRINT #1*
SAVE

Printer Control

LLIST
LPRINT*
TEST

Variables Control

CLEAR*
MEM*
DIM*

Angle Mode Control

DEGREE*
GRAD*
RADIAN*

Other

BEEP*
PASS
RANDOM*
USING*
WAIT*

Text

BASIC
TEXT

* These commands are also BASIC verbs. Their effect as commands is identical to their effect as verbs so they are not described in the command reference section. See the verb reference section for more information.

Verbs

Control and Branching

- CHAIN
- END
- FOR...TO...STEP
- GOSUB
- GOTO
- IF...THEN
- NEXT
- ON...GOSUB
- ON...GOTO
- RETURN
- STOP

Assignment and Declaration

- CLEAR
- DIM
- LET

Input and Output

- AREAD
- CSAVE
- CURSOR
- DATA
- GCURSOR
- GPRINT
- INPUT
- INPUT #
- INPUT #1
- LINE
- LOAD
- LPRINT
- PAUSE
- PRESET
- PRINT
- PRINT #
- PRINT #1
- PSET
- READ
- RESTORE
- USING
- WAIT

Other

- BEEP
- CLOSE
- CLS
- CONSOLE
- DEGREE
- GRAD
- OPEN
- RADIAN
- RANDOM
- REM
- TROFF
- TRON

Functions

Pseudovariables

INKEY$
MEM
MEM$
OPEN$
PI
POINT

String Functions

ASC
CHR$
LEFT$
LEN
MID$
RIGHT$
STR$
VAL

Numeric Functions

ABS
ACS
ASN
ATN
COS
DEG
DMS
EXP
INT
LOG
LN
RND
SGN
SIN
SQR
TAN

COMMANDS

1 **CLOAD**
2 **CLOAD** "filename"

Abbreviations: CLO., CLOA.

See also: CLOAD?, CSAVE, MERGE, PASS

Purpose

The CLOAD command is used to load a program saved on cassette tape. It can only be used with the optional CE-152 or some other cassette recorder.

Use

The first form of the CLOAD command clears the memory of existing programs and loads the first program stored on the tape, starting at the current position.

The second form of the CLOAD command clears the memory, searches the tape for the program whose name is given by "filename", and loads the program.

If the PC-2500 is in PROgram or RUN mode, program memory is loaded from the tape. When the PC-2500 is in the reserve mode, reserve memory is loaded. Care should be taken not to load programs into reserve memory or reserve characters into program memory.

Note: An error results (ERROR 6) if the program exceeds the program/data area.
The program may not fit in the program/data area if this area is used by array variables, simple variables, and the Business Software. Clear the array variables and simple variables with the CLEAR command. Clear the table data created by the Business Software as described on pages 352 and 353. This also applies to the MERGE and CHAIN commands.
Save the data on tape before clearing so that you can use them later.

Examples

CLOAD	Loads the first program from the tape.
CLOAD "PRO3"	Searches the tape for the program named 'PRO3' and loads it.

Notes: 1. The computer cannot identify the stored contents as a program or a reserve. Therefore, if a mode is designated incorrectly, the reserved contents may be transferred to the program area or the program to the reserve area, causing the computer to become inoperative. If this happens, reset the computer by pressing the RESET button on the bottom of the computer.

2. If the designated file name is not retrieved, the computer will continue to search for the file name even after the tape reaches the end. In this case, stop the retrieval function by pressing the [illegible] key. This also applies to MERGE, CHAIN, CLOAD? and INPUT # commands to be described later.
3. If an error occurs during CLOAD or CHAIN command (to be described later) execution, the program stored in the computer will be invalid.

- During loading the mark "*" is shown at the right most digit of the bottom line of the display. The mark "*" will disappear when loading is completed. While a file name is being retrieved, no "*" mark will be displayed as the loading is not started yet.
  (If no file name has been specified, this will occur during reading of the first program.)

1 CLOAD?
2 CLOAD? "filename"

Abbreviations: CLO.?, CLOA.?

See also: CLOAD, CSAVE, MERGE, PASS

Purpose

The CLOAD? command is used to compare a program saved on cassette tape with one stored in memory. It can only be used with the optional CE-152 or some other cassette recorder.

Use

To verify that a program was saved correctly, rewind the cassette tape to the beginning of the program and use the CLOAD? command.
The first form of the CLOAD? command compares the program stored in memory with the first program stored on the tape, starting at the current position.

The second form of the CLOAD? command searches the tape for the program whose name is given by "filename" and then compares it to the program stored in memory.

Examples

CLOAD?	Compares the first program from the tape with the one in memory.
CLOAD? "PRO3"	Searches the tape for the program names "PRO3" and compares it to the one stored in memory.

* The "*" mark is displayed at the bottom right of the display unit when the program is being verified. At the end of verification, the "*" mark disappears and the prompt is displayed.

1 CONT

Abbreviations: C., CO., CON.

See also: RUN, STOP

Purpose

The CONT command is used to continue a program which has been temporarily halted.

Use

When the STOP verb is used to halt a program during execution, the program can be continued by entering CONT in response to the prompt.

When a program is halted using the ON/BRK key, the program can be continued by entering CONT in response to the prompt.

CONT also functions when the program is temporarily interrupted due to a command such as PRINT or GPRINT.

Examples

CONT Continues an interrupted program execution.

1 CSAVE
2 CSAVE "filename"
3 CSAVE, "password"
4 CSAVE "filename", "password"

Abbreviations: CS., CSA., CSAV.

See also: CLOAD, CLOAD?, MERGE, PASS.

Purpose

The CSAVE command is used to save a program to cassette tape. It can only be used with the optional CE-152 or some other cassette recorder,

Use

The first form of the CSAVE command writes all of the programs in memory on to the cassette tape without a specified file name.

The second form of the CSAVE command writes all of the programs in memory on to the cassette tape and assigns the indicated file name.

The third form of the CSAVE command writes all of the programs in memory on to the cassette tape without a specified file name and assigns the indicated password. Programs saved with a password may be loaded by anyone, but only someone who knows the password can list or modify the programs. (See discussion under PASS command).

The fourth form of the CSAVE command writes all of the programs in memory on to the cassette tape and assigns them the indicated file name and password.

If the PC-2500 is in PROgram or RUN mode, program memory is loaded to the tape. When the PC-2500 is in the Reserve mode, reserve memory is loaded.

Examples

CSAVE "PRO3", "SECRET"	Saves the programs now in memory on to the tape under the name 'PRO3', protected with the password 'SECRET'.

1 GOTO expression

Abbreviations: G., GO., GOT.

See also: RUN

Purpose

The GOTO command is used to start execution of a program.

Use

The GOTO command can be used in place of the RUN command to start program execution at the line number specified by the expression.

GOTO differs from RUN in six respects:

1) The value of the interval for WAIT is not reset.
2) The display format established by USING statements is not cleared.
3) Variables and arrays are preserved.
4) PRINT=LPRINT status is not reset.
5) The pointer for READ is not reset.
6) The cursor specification is maintained.
7) The horizontal direction of the graphic cursor is specified with Ø. The setting for the vertical direction is maintained.
8) The serial I/O circuit is not closed.

Execution of a program with GOTO is identical to execution with the DEF key.

Examples

GOTO 1ØØ Begins execution of the program at line 1ØØ.

1 **LIST**
2 **LIST** line number
3 **LIST** "label"

Abbreviations: L., LI., LIS.

See also: LLIST

Purpose

The LIST command is used to display a program.

Use

The LIST command may only be used in the PROgram mode.

* With format (1), the program is displayed from its first line until the display is full.
* With format (2), the program is displayed from the line of the specified line number until the display is full.
  If the line for the specified number does not exist, the program will be displayed from the line with the next largest number which does exist.
* With format (3), the program is displayed from the line written with the specified label until the display is full.
* When programs are merged with the MERGE command, the LIST command functions for the last program.
  However, if the label specified in format (3) does not exist in the last program, it is searched for in sequence from the first program. If the specified label is found, the line containing it is displayed. If a password has been set the LIST command is ignored.

Examples

LIST 1ØØ   Displays line number 1ØØ.

1 **LLIST**
2 **LLIST** expression
3 **LLIST** expression 1, expression 2
4 **LLIST** expression,
5 **LLIST**, expression

Abbreviations: LL., LLI., LLIS.

See also: LIST

Purpose

The LLIST command is used for printing a program on the printer.

Use

* When the serial I/O interface is open due to the OPEN command, the LLIST command outputs the program at the serial I/O interface terminal. (See page 196.) To return the program printing command to the printer, execute the CLOSE command.

The LLIST command may be used in the PROgram or RUN mode.

The first form prints all of the programs in memory.

The second form prints only the program line whose line number is given by expression.

The third form prints the statements from the line number with the nearest line equal to or greater than the value of expression 1 to the nearest line equal to or greater than the value of expression 2. There must be at least two lines between the two numbers.

The forth form prints all program lines beginning with the line whose number is given by the expression.

The fifth form prints all program lines up to, and including, the line whose number is given by the expression.

* When programs are merged with the MERGE command, the LLIST command functions for the last program. To list a program stored earlier, execute.

```
        LLIST "label",
```

If a password has been set the LLIST command is ignored.

Although the number of characters in the printing line is set by the CONSOLE command, an error will result (ERROR 3) if less than 24 columns is set and the LLIST command is executed. Set the printing line to 24 columns or more before executing the LLIST command. (See page 194 for information on the CONSOLE command.)

Examples

LLIST 100, 200   Lists the statements between line numbers 100 and 200.

1 MERGE
2 MERGE "filename"
(effective for the manual operation in the PROgram or RUN mode)

Abbreviations: MER., MERG.

See also: CLOAD

Purpose

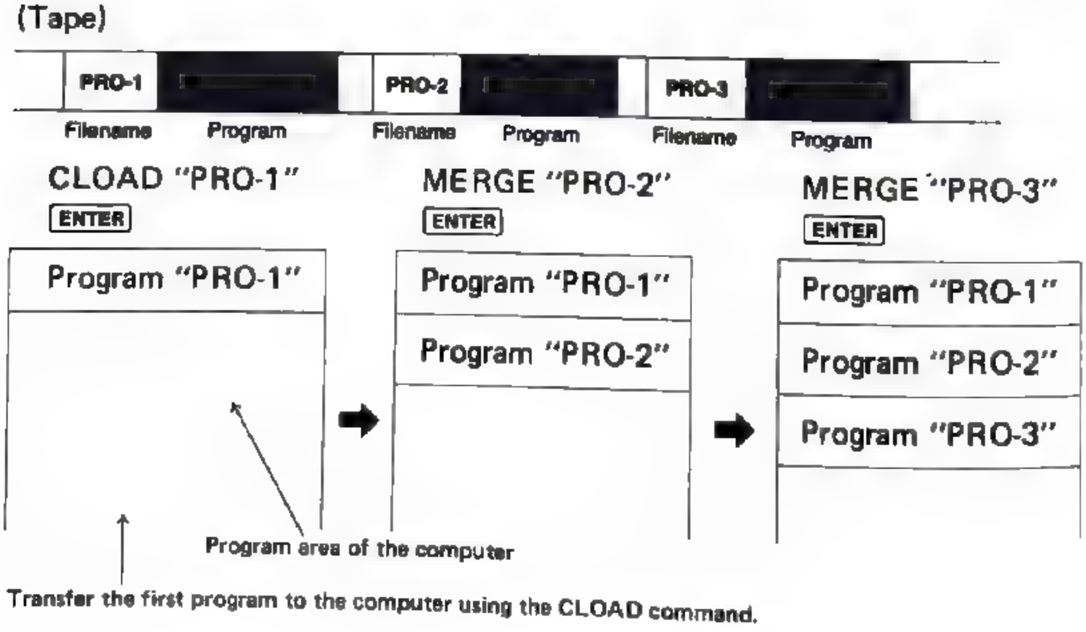
The MERGE command is used to load a program saved on cassette tape and merge it with the program existing in memory.

Use

The MERGE command retains the program already stored in the PC-2500 and then loads a program recorded on the tape. Therefore, several different programs can be stored in the PC-2500 at the same time.

Examples

When programs with file names PRO-1, PRO-2 and PRO-3 are to be stored, PRO-1 is stored using the CLOAD command, and PRO-2 and PRO-3 are transferred to the computer using the MERGE command. The state of the storage is as follows.



Programs loaded using the MERGE command are stored as in the example. The programs are handled as follows by their line numbers.

- If the first line number of the program loaded using the MERGE command is larger than the last line number of the previously loaded program, the two programs are considered to be a single program.
- If the first line number of the program loaded using the MERGE command is smaller than the last line number of the previously loaded program, the two programs are considered separate.
  In the example above, where the line numbers for programs PRO-1, PRO-2, and PRO-3 are 10-200, 50-150, and 160-300 respectively, PRO-1 and PRO-2 are considered separate. PRO-2 and PRO-3 are considered to be a single program with line numbers 50-300.

* Loading programs with the MERGE command may result in 2 or more programs in the PC-2500 with the same line numbers. In this case the executed RUN or GOTO (RUN expression, GOTO expression) is valid only for the last merged program. There will be no way to execute the preceding program(s).
Therefore, put a label at the beginning of the program to be executed and execute using the defined key.
Note, however, that only the last merged program can be edited after the MERGE command has been executed and that the program(s) loaded earlier cannot be edited. Therefore, add the label to the program before merging the next program.

Merging password protected programs

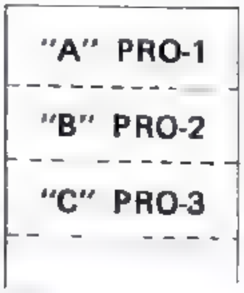
When loading programs with passwords (password protected programs) using the MERGE command, the handling of the programs differ as follows depending on whether the programs within the computer are protected.

When protected
Password protected programs cannot be loaded.

When not protected
If password protected programs are loaded using the MERGE command, all programs within the computer become protected.

When the programs within the computer are protected, programs without passwords become password protected when loaded using the MERGE command.

Executing merged programs



The figure shows the memory when PRO-1 is loaded after which PRO-2 and PRO-3 are loaded using the merge command. If a program is started using RUN or GOTO (RUN expression or GOTO expression), PRO-3 will be executed. On the other hand, if started using RUN "label", GOTO "label", or a defined key, the specified label is searched for from the beginning of PRO-3 within the computer.

If not found in PRO-3, the search proceeds in PRO-1. If also not found in PRO-1, PRO-2 is searched. If the label is found, the program is executed from the labelled line.

Note that since the label is searched for in this manner, if a label used in PRO-1 and PRO-2 is also used in PRO-3, PRO-1 and PRO-2 cannot be executed.

Further, if the LIST command is executed with the label specified as LLIST "A", "C" ENTER , the label is searched for in PRO-3 first.

An error results since "C" appears first.

1 NEW

Abbreviations: none

Purpose

The NEW command is used to clear existing program or reserve memory.

Use

When the NEW command is executed in the PRO mode, the program, array variables, and simple variables within the program/data area are all cleared. However, this function does not clear anything if a password has been set.

When used in the Reserve mode, the NEW command clears all existing reserve memory.
The NEW command is not defined in the RUN mode and will result in an ERROR 9.

Note: Table data created by the Business Software will not be cleared. To clear the table data, follow the method "Delete All Files" given on page 353.

Examples

NEW	Clears program or reserve memory

1 **PASS** "character string"

Abbreviations: PA., PAS.

See also: CSAVE, CLOAD, NEW

Purpose

The PASS command is used to set and cancel passwords.

Use

Passwords are used to protect programs from inspection or modification by other users. A password consists of a character string which is no more than seven characters long. The seven characters must be alphabetic or one of the following special symbols: ! # $ % & ( ) * + – / , . : ; < = > ? @ ' ∧ [ ] –

Once a PASS command has been given, the programs in memory are protected. A password protected program cannot be examined or modified in memory. It cannot be sent to tape or listed with LIST or LLIST, nor is it possible to add or delete program lines. If several programs are in memory and PASS is entered, all programs in memory are protected. The only way to remove this protection is to execute another PASS command with the same password.

Note: When a password with 7 or more characters is declared, only the first 7 characters are valid and are used to set and remove protection.
Press ENTER right after the password.
Writing characters or symbols after the password results in an error and the password cannot be cancelled.
(example) PASS "ABCDEFG":A=123 ENTER → Error

Examples

PASS "SECRET"	Establishes the password 'SECRET' for all programs in memory.

* Regarding the Password When Using the RAM Card
The password for the PC-2500 and for the RAM card are separate.
When the RAM card is mounted, the password set, and the RAM card is removed, the program stored in the RAM card becomes protected. The password for the PC-2500 will not be set at this time.
Further, if the RAM card is mounted after protecting the program within the PC-2500 (i.e. setting the password), the program within the RAM card will not be protected.
(However, the password previously set for the program in the RAM card is valid.)

1 **RUN**
2 **RUN** line number

Abbreviations: R., RU.

See also: GOTO, MERGE

Purpose

The RUN command is used to execute a program in memory.

Use

The first form of the RUN command executes a program beginning with the lowest numbered statement in memory.

The second form of the RUN command executes a program beginning with the specified line number.

* When programs are merged with the MERGE command, the last merged program will be executed with format (1) or "RUN expression" in format (2).

RUN differs from GOTO in eight respects:

1) The value of the interval for WAIT is reset.
2) The display format established by USING statements is cleared.
3) Variables and arrays other than the fixed variables are cleared.
4) PRINT=PRINT status is set.
5) The pointer for READ is reset to the beginning DATA statement.
6) The cursor specification is cleared.
7) The setting for the graphic cursor is cleared to (Ø, 7).
8) Closes the serial I/O circuit (serial port).

Execution of a program with GOTO is identical to execution with the DEF key. In all three forms of program execution FOR/NEXT and GOSUB nesting is cleared.

Examples

RUN 1ØØ	Executes the program which begins at line number 1ØØ.

1 TEST

Abbreviations: TES.

Purpose

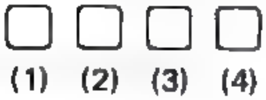
The TEST command is used to check the colors and conditions of the ink in the pens.

Use

Four squares (5 x 5 mm) are drawn in different colors.

Examples

TEST ENTER



(1) Color of position 0 (black)

(2) Color of position 1 (blue)

(3) Color of position 2 (green)

(4) Color of position 3 (red)

VERBS

1 AREAD variable name

Abbreviations: A., AR., ARE., AREA.

See also: INPUT verb and discussion of the use of the DEF key in Chapter 6

Purpose

The AREAD verb is used to read in a single value to a program which is started using the DEF key.

Use

When a program is labelled with a letter, so that it can be started using the DEF key, the AREAD verb can be used to enter a single starting value without the use of the INPUT verb. The AREAD verb must appear on the first line of the program following the label. If it appears elsewhere in the program, it will be ignored. Either a numeric or string variable may be used, but only one can be used per program.

To use the AREAD verb type the desired value in the RUN mode, press the DEF key, followed by the letter which identifies the program. If a string variable is being used, it is not necessary to enclose the entered string in quotes.

Examples

```
10 "X": AREAD N
20 PRINT N^2
30 END
```

Entering "7 DEF X" will produce a display of "49".

Notes:

1. When the display indicates PROMPT (">") at the start of program execution, the designated variable is cleared.
2. When the contents are displayed by a PRINT verb at the start of program execution, the following is stored:

   Example: When the program below is executed;

```
    10 "A": PRINT "ABC", "DEFG"
    20 "S": AREAD A$: PRINT A$
    RUN mode
     DEF   A  → ABC DEFG
     DEF   S  → DEFG
```

- When the display contents are set by PRINT Numeric expression, Numeric expression, Numeric expression...., Numeric expression, or PRINT "String", "String", "String"...., the contents displayed last are stored.
- When the display contents are set by PRINT Numeric expression; numeric expression; Numeric expression...., the contents displayed first (on the extreme left) are stored.
- When the display contents are set by PRINT "String"; "String"; "String"...., the contents of the "String" designated last (on the extreme right) are stored.

1 BEEP <u>expression</u>

Abbreviations: B., BE., BEE.

Purpose

The BEEP verb is used to produce an audible tone.

Use

The BEEP verb causes the PC-2500 to emit one or more audible tones at 4 kHz. The number of beeps is determined by the expression, which must be numeric. (Positive number less than 9.999999999E+99) The expression is evaluated, but only the integer part is used to determine the number of beeps.

BEEP may also be used as a command using numeric literals and predefined variables. In this case the beeps occur immediately after the ENTER key is pressed.

Examples

```
1Ø A=5 : B$="9"
2Ø BEEP 3        Produces 3 beeps.
3Ø BEEP A        Produces 5 beeps.
4Ø BEEP(A+4)/2   Produces 4 beeps.
5Ø BEEP B$       This is illegal and will produce an ERROR 9 message.
6Ø BEEP –4       Produces no beeps, but does not produce an error message.
```

1 CHAIN
2 CHAIN expression
3 CHAIN "filename"
4 CHAIN "filename", expression

Abbreviations: CH., CHA., CHAI.

See also: CLOAD, CSAVE, and RUN

Purpose

The CHAIN verb is used to start execution of a program which has been stored on cassette tape. It can only be used in connection with the optional CE-152 or some other cassette recorder.

Use

To use the CHAIN verb one or more programs must be stored on a cassette. Then, when the CHAIN verb is encountered in a running program, a program is loaded from the cassette and executed.

The first form of CHAIN loads the first program found on the tape and begins execution with the lowest line number in the program. The effect is the same as having entered CLOAD and RUN when in the RUN mode.

The second form of CHAIN loads the first program found on the tape and begins execution with the line number specified by the expression.

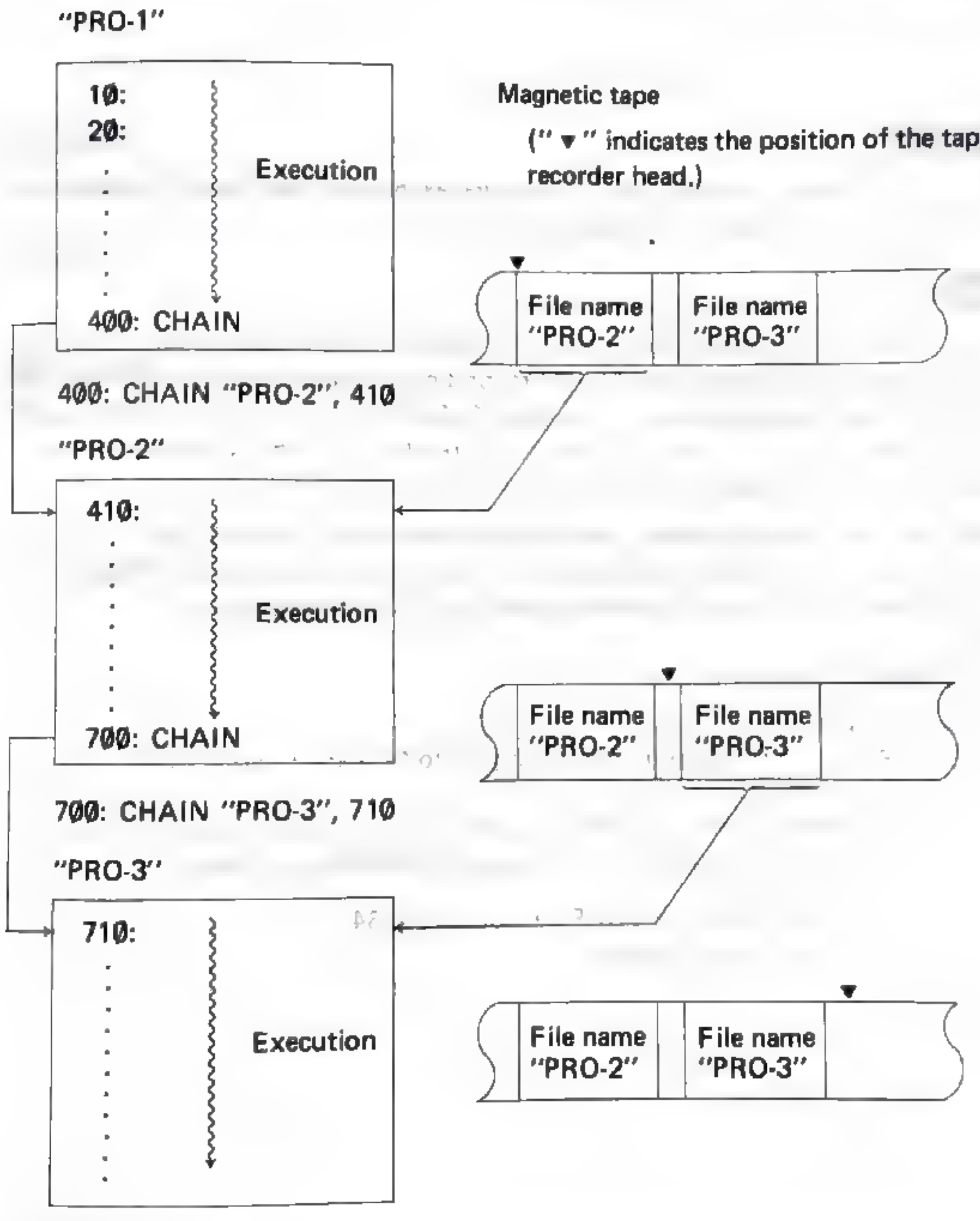
The third form of CHAIN searches the tape for the program whose name is indicated by "filename", loads the program, and begins execution with the lowest line number.

The fourth form of CHAIN will search the tape for the program whose name is indicated by filename, load the program, and begin execution with the line number indicated by the expression.

Examples

1Ø CHAIN	Loads the first program found on the tape and begins execution with the lowest line number.
2Ø CHAIN "PRO-2", 48Ø	Searches the tape for a program named PRO-2, loads it, and begins exectuion with line number 48Ø.

For example, let's assume you have three program sections named PRO-1, PRO-2, PRO-3. Each of these sections ends with a CHAIN statement.



```
4ØØ: CHAIN "PRO-2", 41Ø
```

```
7ØØ: CHAIN "PRO-3", 71Ø
```

```
99Ø: END
```

During execution, when the computer encounters the CHAIN statement, the next section is called into memory and executed. In this manner, all of the sections are eventually run.

1 CLEAR

Abbreviations: CL., CLE., CLEA.

See also: DIM

Purpose

The CLEAR verb is used to erase all variables which have been used in the program and to reset all preallocated variables to zero or null.

Use

The CLEAR verb recovers space which is being used to store variables. This might be done when the variables used in the first part of a program are not required in the second part and available space is limited. CLEAR may also be used at the beginning of a program when several programs are resident in memory and you want to clear out the space used by execution of prior programs.

Clear does not free up the space used by the variables A–Z, A$–Z$, or A(1)–A(26) (without DIM declaration) since they are permanently assigned (see Chapter 4). CLEAR does reset numeric variables to zero and string variables to null.

Examples

```
10 A=5: DIM C(5)
20 CLEAR           Frees up the space assigned to C( ) and resets A to zero.
```

Note: If you want to use more than 27 subscripts with variable array A( ), execute the statements shown below following the CLEAR statement.

```
100   CLEAR: POKE &6D89, 0, 64, 64
110   CALL &84DE
```

1 CLS

Abbreviations: none

See also: CURSOR

Purpose

The CLS command clears the display.

Use

Clears the display and returns the display start position to 0.

Examples

```
10:WAIT 3
20:INPUT A$
30:FOR B=0 TO 23
40:CLS
50:CURSOR B, 1
60:PRINT A$
70:NEXT B
80:CLS
90:END
```

This program displays the entry while moving it from left to right on the display unit (from the upper line to the lower line). Each time the FOR-NEXT loop of lines 30–70 is executed, the display is cleared with the CLS command, and display start position is shifted with the CURSOR command, and the contents of A$ are displayed with the PRINT command. By writing and clearing the display in this manner, the display can be made to appear to move. (Delete line 40 and execute. Note the difference.)

1 CURSOR expression 1, expression 2
2 CURSOR expression
3 CURSOR

Abbreviations: CU., CUR., CURS., CURSO.

See also: GCURSOR, CLS, INPUT, PRINT, PAUSE

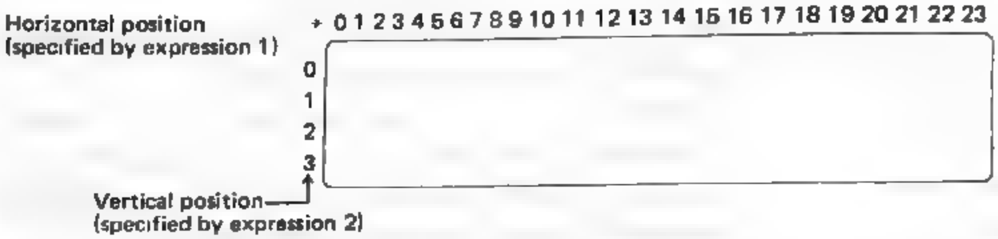
Purpose

Specifies the display start position in column units.

Use

Format (1) and format (2) specify the display start position in units of a character position for the contents displayed by the PRINT command, PAUSE command, etc.

* The display position is specified as follows using format (1).

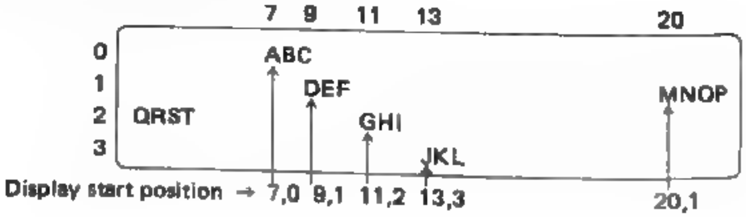


A position on the display unit is specified by its horizontal and vertical positions. The values of expression 1 and expression 2 specify the horizontal and vertical position, respectively.
Therefore, be sure that the value of expression 1 is in the range of Ø–23 and the value of expression 2 is in the range of Ø–3.

Examples

```
 5 CLS
1Ø CURSOR 7,Ø:PRINT"ABC"
2Ø CURSOR 9,1:PRINT"DEF"
3Ø CURSOR 11,2:PRINT"GHI"
4Ø CURSOR 13,3:PRINT"JKL"
5Ø CURSOR 20,1:PRINT"MNOPQRST"
```

Executing the program will display the following.



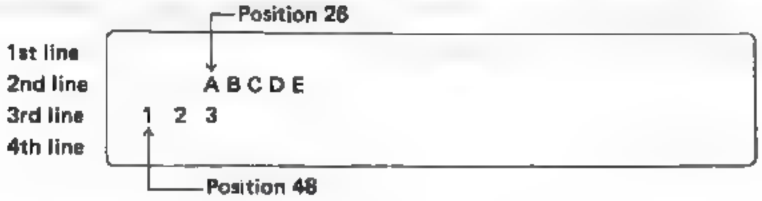
* The display position is specified as follows using format (2).

```
1st line | 0  1  2  3  . . . . . . . . . . . . . . . . . . . . . . . 22 23
2nd line | 24 25 26 . . . . . . . . . . . . . . . . . . . . . . . . . 46 47
3rd line | 48 49 50 . . . . . . . . . . . . . . . . . . . . . . . . . 69 70
4th line | 71 72 73 . . . . . . . . . . . . . . . . . . . . . . . . . . . 95
```

As shown, the positions on the display are assigned numbers from Ø to 95, starting from the top left of the display unit to the bottom right. The value of the expression in format (2) specifies the number of the position for the display start position. Therefore, be sure that the value of the expression in format (2) is within the range of Ø–95. Exceeding this range results in a error (ERROR 9).

Example

```
 5 CLS
1Ø CURSOR 48:PRINT 123
2Ø CURSOR 26:PRINT "ABCDE"
```



* Format (3) clears the display start position.

The characters in front of and behind those displayed by the PRINT command or PAUSE command after the display start position was specified with the CURSOR command are retained. This feature can be used in many ways to change only a portion of the display. To clear the display, use the CLS command.

* The display start position specified in the CURSOR command can also be used for the INPUT command. However, after the INPUT command is executed, the display start position will still be specified at this position.
  Therefore, note that the characters will be displayed from the same position if a PRINT command is executed next.

Example

```
  1Ø CLS
  2Ø CURSOR Ø,2
┌→3Ø INPUT"DATA=";A
├→4Ø PRINT A/2
└─The position specified in line 2Ø applies for lines 3Ø and 4Ø.
```

```
  1Ø CLS
  2Ø CURSOR Ø,2
  3Ø INPUT"DATA=";A
┌→35 CURSOR
│ 4Ø PRINT A/2
└─The position specified in line 2Ø is cleared.
```

* If the displayed characters overflow the screen, the screen is moved up (scrolled) to display all the characters even though the display start position was specified with the CURSOR command.

1 **DATA** expression list
Where: expression list is: expression
or: expression, expression list

Abbreviations: DA., DAT.

See also: READ, RESTORE

Purpose

The DATA verb is used to provide values for use by the READ verb.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR . . . NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

DATA statements have no effect if encountered in the course of regular execution of the program, so they can be inserted wherever it seems appropriate. Many programmers like to include them immediately following the READ which uses them. If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Example

```
1Ø DIM B(1Ø)              Sets up an array.
2Ø WAIT 128
3Ø FOR H=1 TO 1Ø
4Ø READ B(H)              Loads the values from the DATA statement into B(  )
5Ø PRINT B(H)               B(1) will be 1Ø, B(2) will be 2Ø, B(3) will be 3Ø, etc.
6Ø NEXT H
7Ø DATA 1Ø,2Ø,3Ø,4Ø,5Ø,6Ø
8Ø DATA 7Ø,8Ø,9Ø,1ØØ
9Ø END
```

1 DEGREE

Abbreviations: DE., DEG., DEGR., DEGRE.

See also: GRAD and RADIAN

Purpose

The DEGREE verb is used to change the form of angular values to decimal degrees.

Use

The PC-2500 has three forms for representing angular values — decimal degrees, radians and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The DEGREE function changes the form for all angular values to decimal degree form until a GRAD or RADIAN verb is used. The DMS and DEG functions can be used to convert decimal degrees to degree, minute, second form and vice versa.

Example

```
1Ø DEGREE
2Ø X=ASN 1          X now has a value of 9Ø, i.e. 9Ø degrees, the Arcsine of 1.
3Ø PRINT X
```

1 **DIM** dim list

Where: dim list	is:	dimension spec.
	or:	dimension spec., dim list
and: dimension spec.	is:	numeric dim spec.
	or:	string dim spec.
and: numeric dim spec	is:	numeric name (size)
and: string dim spec	is:	string name (dims)
	or:	string name (dims)*len
and: numeric name	is:	valid numeric variable name
and: string name	is:	valid string variable name
and: dims	is:	size
	or:	size, size
and: size	is:	number of elements
and: len	is:	length of each string in a string array

Abbreviations: D., DI.

Purpose

The DIM verb is used to reserve space for numeric and string array variables.

Use

Except for an array of the form A( ) or A$( ), a DIM verb must be used to reserve space for any array variable.

The maximum number of dimensions in any array is two; the maximum size of any one dimension is 255. In addition to the number of elements specified in the dimension statement, one additional "zeroeth" element is reserved. For example, DIM B(3) reserves B(Ø), B(1), B(2) and B(3). In two dimensional arrays there is an extra "zeroeth" row and column.

In string arrays one specifies the size of each string element in addition to the number of elements. For example, DIM B$(3)*12 reserves space for 4 strings which are each a maximum of 12 characters long. If the length is not specified each string can contain a maximum of 16 characters.

When a numeric array is dimensioned, all values are initially set to zero; in a string array the values are set to null.

For the array A and A$ DIM declaration, refer to the paragraph discussing variables.

Array variables can be cleared (or set undefined) with the CLEAR command. When the program is started using the RUN command, array variables are automatically cleared.

The variable name once declared cannot be declared again. When a program once executed is executed again with the GOTO command on using the DEF key, the same variable name as formerly declared will be declared again if the line with the DIM command is executed. In this case, clear the array variable with the CLEAR command and then declare it again.

Example

1Ø DIM B(1Ø) — Reserves space for a numeric array with 11 elements.

2Ø DIM C$(4, 4)*1Ø — Reserves space for a two dimensional string array with 5 rows and 5 columns; each string will be a maximum of 1Ø characters.

1 END

Abbreviations: E., EN.

Purpose

The END verb is used to signal the end of a program and close the serial I/O circuit.

Use

When multiple programs are loaded into memory at the same time a mark must be included to indicate where each program ends so that execution does not continue from one program to another. This is done by including an END verb as the last statement in the program.

Examples

```
1Ø PRINT "HELLO"
2Ø END
3Ø PRINT "GOODBYE"
4Ø END
```

With these programs in memory a 'RUN 1Ø' prints 'HELLO', but not 'GOODBYE'. 'RUN 3Ø' prints 'GOODBYE'.

1 **FOR** numeric variable=expression 1 **TO** expression 2
2 **FOR** numeric variable=expression 1 **TO** expression 2
STEP expression 3

Abbreviations: F. and FO.; STE.

See also: NEXT

Purpose

The FOR verb is used in combination with the NEXT verb to repeat a series of operations a specified number of times.

Use

The FOR and the NEXT verbs are used in pairs to enclose a group of statements which are to be repeated. The first time this group of statements is executed the loop variable (the variable named immediately following the FOR) has the value of expression 1.

When execution reaches the NEXT verb the loop variable is increased by the step size and then this value is tested against expression 2. If the value of the loop variable is less than or equal to expression 2, the enclosed group of statements is executed again, starting with the statement following the FOR. In the first form the step size is 1; in the second form the step size is given by expression 3. If the value of the loop variable is greater than expression 2, execution continues with the statement which immediately follows the NEXT. Because the comparison is made at the end, the statements within a FOR/NEXT pair are always executed at least once.

Expression 1, expression 2, and expression 3 must be in the range of −9.999999999E99 ~9.999999999E99. When expression 3 is zero the FOR/NEXT loop will be infinite.

The loop variable may be used within the group of statements, for example as an index to an array, but care should be taken in changing the value of the loop variable.

Programs should be written so that they never jump from outside a FOR/NEXT pair to a statement within a FOR/NEXT pair. Similarly, programs must never leave a FOR/NEXT pair by jumping out. Always exit a FOR/NEXT loop via the NEXT statement. To do this, set the loop variable to a value higher than expression 2.

The group of statements enclosed by a FOR/NEXT pair can include another pair of FOR/NEXT statements which use a different loop variable as long as the enclosed pair is completely enclosed; i.e., if a FOR statement is included in the group, the matching NEXT must also be included. FOR/NEXT pairs may be "nested" up to five levels deep.

Examples

```
1Ø FOR H=1 TO 5
2Ø PRINT H
3Ø NEXT H
```

This group of statements prints the numbers 1, 2, 3, 4, 5.

```
4Ø FOR N=1Ø TO Ø STEP–1
5Ø PRINT N
6Ø NEXT N
```

This group of statements counts down 1Ø, 9, 8, 7, 6, 5, 4, 3, 2, 1, Ø.

```
7Ø FOR N=1 TO 1Ø
8Ø X=1
9Ø FOR F=1 TO N
1ØØ X=X*F
11Ø NEXT F
12Ø PRINT X
13Ø NEXT N
```

This group of statements computes and prints N factorial for the numbers from 1 to 1Ø.

Note: If execution jumps out of the FOR-NEXT loop, a nesting error (ERROR 5) will occur if a subsequent FOR-NEXT loop is encountered.

1 **GOSUB** expression

Abbreviations: GOS., GOSU.

See also: GOTO, ON . . . GOSUB, ON . . . GOTO, RETURN

Purpose

The GOSUB verb is used to execute a BASIC subroutine.

Use

When you wish to execute the same group of statements several times in the course of a program or use a previously written set of statements in several programs, it is convenient to use the BASIC capability for subroutines using the GOSUB and RETURN verbs.

The group of statements is included in the program at some location where they are not reached in the normal sequence of execution. A frequent location is following the END statement which marks the end of the main program. At those locations in the main body of the program where subroutines are to be executed, include a GOSUB statement with an expression which indicates the starting line number of the subroutine. The last line of the subroutine must be a RETURN. When GOSUB is executed, the PC-2500 transfers control to the indicated line number and processes the statements until a RETURN is reached. Control is then transferred back to the statement following the GOSUB.

A subroutine may include a GOSUB. Subroutines may be "nested" in this fashion up to 1Ø levels deep.

The expression in a GOSUB statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON . . . GOSUB structure for choosing different subroutines at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Examples

```
1Ø GOSUB 1ØØ
2Ø END
1ØØ PRINT "HELLO"
11Ø RETURN
```

When this program is run it prints the word 'HELLO'.

1 **GOTO** expression

Abbreviations: G., GO., GOT.

See also: GOSUB, ON . . . GOSUB, ON . . . GOTO

Purpose

The GOTO verb is used to transfer control to a specified line number.

Use

The GOTO verb transfers control from one location in a BASIC program to another location. Unlike the GOSUB verb, GOTO does not "remember" the location from which the transfer occurred.

The expression in a GOTO statement may not include a comma, e.g., 'A(1, 2)' cannot be used. Since there is an ON . . . GOTO structure for choosing different destinations at given locations in the program, the expression usually consists of just the desired line number. When a numeric expression is used, it must evaluate to a valid line number, i.e., 1 to 65279, or an ERROR 4 will occur.

Well designed programs usually flow simply from beginning to end, except for subroutines executed during the program. Therefore, the principal use of the GOTO verb is as a part of an IF . . . THEN statement.

Examples

```
1Ø INPUT A$
2Ø IF A$="Y" THEN GOTO 5Ø
3Ø PRINT "NO"
4Ø GOTO 6Ø
5Ø PRINT "YES"
6Ø END
```

This program prints 'YES' if a 'Y' is entered and prints 'NO' if anything else is entered.

1 GRAD

Abbreviations: GR., GRA.

See also: DEGREE and RADIAN

Purpose

The GRAD verb is used to change the form of angular values to gradient form.

Use

The PC-2500 has three forms for representing angular values — decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The GRAD function changes the form for all angular values to gradient form until a DEGREE or RADIAN verb is used. Gradient form represents angular measurement in terms of percent gradient, i.e., a 45° angle is a 5Ø^{g} gradient.

Examples

```
1Ø GRAD
2Ø X=ASN 1
3Ø PRINT X
```

X now has a value of 1ØØ, i.e., a 1ØØ^{g} gradient, the Arcsine of 1.

1 **IF** condition **THEN** statement
2 **IF** condition statement

Abbreviations: none for IF, T., TH., THE.

Purpose

The IF ... THEN verb pair is used to execute or not to execute a statement depending on conditions at the time the program is run.

Use

In the normal running of BASIC programs, statements are executed in the sequence in which they occur. The IF ... THEN verb pair allows decisions to be made during execution so that a given statement is executed only when desired. When the condition part of the IF statement is true, the statement is executed; when it is false, the statement is skipped.

The condition part of the IF statement can be any relational expression as described in Chapter 4. It is also possible to use a numeric expression as a condition, although the intent of the statement will be less clear. Any expression which evaluates to zero or a negative number is considered false; any which evaluates to a positive number is considered true.

The statement which follows the THEN may be any BASIC statement, including another IF ... THEN. If it is a LET statement, the LET verb itself must appear. Unless the statement is an END, GOTO, or ON ... GOTO, the statement following the IF ... THEN statement is the next one executed regardless of whether the condition is true.

The two forms of the IF statement are identical in action, but the first form is clearer.

Examples

```
1Ø INPUT "CONTINUE? "; A$
2Ø IF A$="YES" THEN GOTO 1Ø
3Ø IF A$="NO" THEN GOTO 6Ø
4Ø PRINT "YES OR NO, PLEASE"
5Ø GOTO 1Ø
6Ø END
```

This program continues to ask "CONTINUE? ' as long as 'YES' is entered; it stops if 'NO' is entered, and complains otherwise.

1 **INPUT** input list

Where: input list	is: input group
	or: input group, input list
and: input group	is: var list
	or: prompt, var list
	or: prompt; var list
and: var list	is: variable
	or: variable, var list
and: prompt	is: any string constant

Abbreviations: I., IN., INP., INPU.

See also: INPU #, READ, CURSOR, PRINT

Purpose

The INPUT verb is used to enter one or more values from the keyboard.

Use

When you want to enter different values each time a program is run, use the INPUT verb to enter these values from the keyboard.

In its simplest form the INPUT statement does not include a prompt string; instead a question mark is displayed on the left edge of the display. A value is then entered, followed by the ENTER key. This value is assigned to the first variable in the list. If other variables are included in the same INPUT statement, this process is repeated until the list is exhausted.

If a prompt is included in the INPUT statement, the process is exactly the same except that, instead of the question mark, the prompt string is displayed at the left edge of the display. If the prompt string is followed by a semicolon, the cursor is positioned immediately following the prompt. If the prompt is followed by a comma, the prompt is displayed. Then when a key is pressed, the display is cleared and the first character of the input is displayed at the left edge.

When a prompt is specified and there is more than one variable in the list following it, the second and succeeding variables are prompted with the question mark. If a second prompt is included in the list, it is displayed for the variable which immediately follows it.

When the display starting position has been specified using the CURSOR command before executing the INPUT command, the input prompt or "? " will be displayed from that position.

If the ENTER key is pressed and no input is provided, the variable retains the value it had before the input statement and any further instructions on the same line are ignored.

Examples

1Ø INPUT A	Clears the display and puts a question mark at the left edge.
2Ø INPUT "A=";A	Displays 'A=' and waits for input data.
3Ø INPUT "A=",A	Displays'A='. When data is input 'A=' disappears and the data is displayed starting at left edge.
4Ø INPUT "X=? ";X,"Y=? ";Y	Displays 'X=? ' and waits for first input. After ENTER is pressed, display is cleared and 'Y=? ' is displayed at left edge.

Notes:

Clear any errors during input of data in response to the INPUT command by pressing the CLS key and then input the correct data. When playing back using the ← or → key, unnecessary displays may appear.

The data input proceeds normally even if this occurs.

The last answer feature is not used when the INPUT verb used.

1 INPUT # var list
2 INPUT # "filename"; var list

Where: var list is: variable
or: variable, var list

Abbreviations: I. #, IN. #, INP. #, INPU. #

See also: INPUT, PRINT #, READ

Purpose

The INPUT # verb is used to enter values from the cassette tape.

Use and Examples

The following variable types can be specified in the INPUT # statement:
(1) Fixed variables – A, B, C, A(7), D*, A(2Ø)*, etc.
(2) Simple variables – AA, B3, CP$, etc.
(3) Array variables – S(*), HP(*), K$(*), etc.

1) Transferring data to fixed variables

To transfer data from tape to fixed variables, specify the variable names in the INPUT # statement.

```
INPUT # "DATA 1" ; A, B, X, Y
```

This statement transfers data from the cassette file named "DATA 1" to the variables A, B, X, and Y in that order.

To fill all the available fixed variables and, if defined, extended variables (A(27) and beyond) with data transferred from tape, specify the first variable with an asterisk (*) subscripted to it.

```
INPUT # "D-2"; D*
```

This statement transfers the contents of the tape file "D-2" to variables D through Z and to A(27) and beyond.

```
INPUT # A(1Ø)* (without DIM declaration)
```

This statement transfers the data of the first file found after the tape was started, to the variables A(1Ø) and beyond (to J through Z and A(27) and beyond).

Note 1. If an array named A is already defined by the DIM statement, it is not possible to define subscripted fixed variables in the form of A( ).

Note 2. Data transfer to fixed variables and extended variables (A(27) and beyond) will continue until the end of the source data file on the tape is reached.

2) Data transfer to simple variables

Data in a tape file can be transferred to simple variables by specifying the desired simple variable names in the INPUT # statement.

```
INPUT # "DM-1"; AB, Y1, XY$
```

This statement transfers data from the tape file named "DM-1" to simple variables AB, Y1, and XY$.

Note 1. Numeric data must be transferred to numeric simple variables, and character data must be transferred to simple character variables. Cross-transfer is not allowed.

Note 2. Locations for simple variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use assignment statements to reserve the locations for simple variables.

```
AA=Ø ENTER
B1$="A" ENTER
INPUT # AA, B1$ ENTER
```

Use appropriate numeric values or characters in assignment statements to reserve locations for variables.

3) Data transfer to array variables

To transfer data from a tape file to array variables, specify the array name in the INPUT # statement in the form of array name(*).

```
5Ø DIM B(5)
6Ø INPUT # "DS-4"; B(*)
```

This statement transfers data from the tape file named "DS-4" to the variables (B(Ø) through B (5)) in array B.

Note 1. Numeric data must be transferred to numeric array variables with the same length as that of the data, character data must be transferred to character array variables with the same length as that of the data. If this rule is not observed, an error will result.

Note 2. Locations for array variables must be set aside in the program data area before the INPUT # statement is executed. If not, an error will result. Use the DIM statement to define the array in advance.

—CAUTION—

If the number of variables specified in the INPUT # statement does not agree with the amount of data recorded on the tape, the following will happen:

* If the number of pieces of data recorded on the tape file (to be transferred) is greater than the number of specified variables, data transfer will be performed to the last variable, and the remaining data will be ignored.

* If the number of pieces of data recorded in the tape file (to be transferred) is smaller than the number of specified variables, all the file data will be transferred to the variables to the end of the file, and the remaining variables will maintain their previous contents. In this case, however, the computer will continue to wait for data transfer from the tape. To halt this state, you should operate the BRK key.

* If the INPUT # statement is executed with no variable name specified in it, an error (ERROR 1) will result.

1 **LET** variable=expression
2 variable=expression

Abbreviations: LE.

Purpose

The LET verb is used to assign a value to a variable.

Use

The LET verb assigns the value of the expression to the designated variable. The type of the expression must match that of the variable, i.e. only numeric expressions can be assigned to numeric variables and only string expressions can be assigned to string variables. In order to convert from one type to the other, one of the explicit type conversion functions, STR$ or VAL, must be used.

The LET verb may be omitted in all LET statements except those which appear in the THEN clause of an IF . . . THEN statement. In this one case the LET verb must be used.

Examples

10 H=10	Assigns the value 10 to H.
20 A=5*H	Assigns the value 50 to A.
30 X$=STR$ (A)	Assigns the value '50' to X$
40 IF H>=10 THEN LET Y$=X$+".00"	Assigns the value '50.00' to Y$.

1 **LPRINT** print expr
2 **LPRINT** print expr, print expr, . . ., print expr
3 **LPRINT** print list
4 **LPRINT** print list;
5 **LPRINT**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: LP., LPR., LPRI., LPRIN.

See also: PAUSE, PRINT, USING, and WAIT

Purpose

The LPRINT verb is used to print information on the printer.

Use

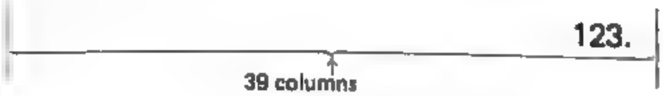
When the serial I/O interface is open due to the OPEN command, the LPRINT command outputs the program at the serial I/O interface terminal. (See page 199.) To return the program printing command to the built-in printer, execute the CLOSE command.

The LPRINT verb is used to print prompting information, results of calculations, etc.

In format (1), the numeric value is right justified and the characters are printed from the left end of the paper within the range of column specified in the CONSOLE command (usually 39 columns). (See page 194 for information on the CONSOLE command.)

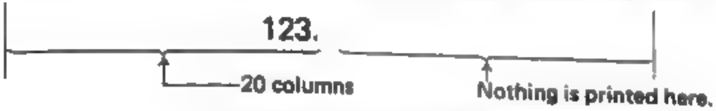
Example: LPRINT 123 ENTER is executed with standard character size (size b)

1. When 39 columns is specified per printing line using the CONSOLE command



(Characters are printed within this range.)

2. When 20 columns is specified per printing line using the CONSOLE command



(Characters are printed within this range.)

In format (2), the number of print columns is delimited into groups of 12 columns. The specified values are printed in sequence. In other words, the first specified value is printed within the first group of 12 columns, the second specified value within the second group, the third specified value within the third group, and the fourth specified value within the fourth group.

Numeric values are printed at the right end of the 12 column range, while character values (string value) are printed starting at the left. If the value to be printed exceeds 12 columns, numeric values are printed with the least significant digit(s) of the decimal fraction truncated so that the value is within 12 digits, and characters are printed from the first 12 (from the left).

Note 1: The number of values (items) specified in format (2) must be within 2–8.
Note 2: If the USING clause has been specified, when format (2) is used, the specification will be cleared and printing will be performed in the form as shown above. The values are printed from the left edge of the paper.

In format (3), the values are printed from the left edge of the paper. If the value to be printed exceeds the number of columns specified by the CONSOLE command, a new line is automatically performed. Up to a maximum of 96 characters can be printed. An error occurs if the 96th column is in the middle of a numeric value.

In format (4), at the specified printing value, the value specified in the LPRINT command to be executed next will be printed in succession.

In format (5), no printing occurs but the paper is fed one line.

See page 73 for the various printer settings.

Examples

```
10 A=10:B=20:X$="ABCDE":Y$="XYZ"
20 LPRINT A
30 LPRINT X$
40 LPRINT A,B,X$,Y$
50 LPRINT X$;A;B
60 LPRINT
70 LPRINT A*B;
80 LPRINT Y$
```

.1 **NEXT** numeric variable

Abbreviations: N., NE., NEX.

See also: FOR

Purpose

The NEXT verb is used to mark the end of a group of statements which are being repeated in a FOR/NEXT loop.

Use

The use of the NEXT verb is described under FOR. The numeric variable in a NEXT statement must match the numeric variable in the corresponding FOR.

Examples

```
1Ø FOR H=1 TO 1Ø
2Ø PRINT H
3Ø NEXT H
```

Prints the numbers from 1 to 1Ø each time ENTER is pressed.

1 **ON** expression **GOSUB** expression list

Where: expression list is: expression
or: expression, expression list

Abbreviations: O.; GOS., GOSU.

See also: GOSUB, GOTO, ON ... GOTO

Purpose

The ON ... GOSUB verb is used to execute one of a set of subroutines depending on the value of a control expression.

Use

When the ON ... GOSUB verb is executed the expression between ON and GOSUB is evaluated and reduced to an integer. If the value of the integer is 1, the first subroutine in the list is executed as in a normal GOSUB. If the expression is 2, the second subroutine in the list is executed, and so forth. After the RETURN from the subroutine execution proceeds with the statement which follows the ON ... GOSUB.

If the expression is zero, negative, or larger than the number of subroutines provided in the list, no subroutine is executed and execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOSUB. The PC-2500 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
1Ø INPUT A
2Ø ON A GOSUB 1ØØ, 2ØØ, 3ØØ
3Ø END
1ØØ PRINT "FIRST"
11Ø RETURN
2ØØ PRINT "SECOND"
21Ø RETURN
3ØØ PRINT "THIRD"
31Ø RETURN
```

An input of 1 prints "FIRST"; 2 prints "SECOND"; 3 prints "THIRD". Any other input does not produce any print.

1 ON expression GOTO expression list

Where: expression list is: expression
                        or: expression, expression list

Abbreviations: O.; G., GO., GOT.

See also: GOSUB, GOTO, ON ... GOSUB

Purpose

The ON ... GOTO verb is used to transfer control to one of a set of locations depending on the value of a control expression.

Use

When the ON ... GOTO verb is executed the expression between ON and GOTO is evaluated and reduced to an integer. If the value of the integer is 1, control is transferred to the first location in the list. If the expression is 2, control is transferred to the second location in the list; and so forth.

If the expression is zero, negative, or larger than the number of locations provided in the list, execution proceeds with the next line of the program.

NOTE: Commas may not be used in the expressions following the GOTO. The PC-2500 cannot distinguish between commas in expressions and commas between expressions.

Examples

```
10  INPUT A
20  ON A GOTO 100,200,300
30  GOTO 900
100  PRINT "FIRST"
110  GOTO 900
200  PRINT "SECOND"
210  GOTO 900
300  PRINT "THIRD"
310  GOTO 900
900  END
```

An input of 1 prints 'FIRST; 2 prints 'SECOND'; 3 prints 'THIRD'. Any other input does not produce any print.

1 **PAUSE** print expr
2 **PAUSE** print expr, print expr, ..., print expr
3 **PAUSE** print list
4 **PAUSE** print list;
5 **PAUSE**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: PAU., PAUS.

See also: LPRINT, PRINT, CURSOR, USING, and WAIT

Purpose

The PAUSE verb is used to print information on the display for a short period.

Use

The PAUSE verb is used to display prompting information, results of calculations, etc. The operation of PAUSE is identical to PRINT except that after PAUSE the PC-2500 waits for short preset interval of about .85 seconds and then continues execution of the program without waiting for the ENTER key or the WAIT interval.

The first form of the PAUSE statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

However, when the display starting position is specified using format (4) or the CURSOR command, the display starts from that position.
In format (2), the display unit is divided into groups of 12 columns. The values are displayed, in sequence, from the first specified value. The numeric value of an expression is displayed from the right end of the 12 column group and characters are displayed from the left side.

- The number of the values (items) specified in format (2) must be within 2–8.
- If the specified value exceeds 12 columns, the following is performed.
  1) When the numeric value exceeds 12 digits (when the decimal fraction in the exponential display is 7 digits or more), the least significant digits are truncated.
  2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format (3), the specified value is displayed continuously from the left side of the display. However, if the display starting position has been specified using format (4) or the CURSOR command, the display starts from that position.

Note: If the value to be displayed in format (3) exceeds 96 columns, the excess portion is not displayed. When the value to be displayed does exceed 96 columns, an error (ERROR 6) occurs if the 96 column is in the middle of a numeric value.

In format (4), the specified value is displayed from the left side of the display. The column following the end of this displayed value is specified as the display starting position for display commands such as for the next PRINT command.
In format (5), the previously displayed value is displayed as is.

Examples

```
1Ø A=1Ø:B=2Ø:X$="ABCDEF":
   Y$="XYZ"

                              Display

2Ø PAUSE A                    |                        1Ø.|

3Ø PAUSE X$                   | ABCDEF                    |

4Ø PAUSE X$, Y$, A, B         | ABCDEF          XYZ       |
                              |             1Ø.        2Ø.|

5Ø PAUSE Y$;X$;               | XYZABCDEF                 |

6Ø PAUSE A*B                  | XYZABCDEF2ØØ.             |
```

1 **PRINT** print expr
2 **PRINT** print expr, print expr, print expr, print expr
3 **PRINT** print list
4 **PRINT** print list;
5 **PRINT**
6 **PRINT=LPRINT**
7 **PRINT=PRINT**

Where: print list is: print expr
or: print expr; print list
and: print expr is: expression
or: USING clause; expression

The USING clause is described separately under USING

Abbreviations: P., PR., PRI., PRIN.

See also: LPRINT, PAUSE, CURSOR, USING, and WAIT

Purpose

The PRINT verb is used to print information on the display or on the printer.

Use

The PRINT verb is used to display prompting information, results of calculations, etc. The first form of the PRINT statement displays a single value. If the expression is numeric, the value is printed at the far right end of the display. If it is a string expression, the display is made starting at the far left.

However, when the display starting position is specified using format (4) or the CURSOR command, the display starts from that position.

In format (2), the display unit is divided into groups of 12 columns. The values are displayed, in sequence, from the first specified value. The numeric value of an expression is displayed from the right end of the 12 column group and characters are displayed from the left end.

- The number of the values (items) specified in format (2) must be within 2–8.
- If the specified value exceeds 12 columns, the following is performed.
  1) When the numeric value exceeds 12 digits (when the decimal fraction in the exponential display is 7 or more), the least significant digits are truncated.
  2) When the characters exceed 12 columns, only the first 12 characters (from the left) are displayed.

In format (3), the specified value is displayed continuously from the left side of the display. However, if the display starting position has been specified using format (4) or the CURSOR command, the display starts from that position.

Note: If the value to be displayed in format (3) exceeds 96 columns, the excess portion is not displayed. When the value to be displayed does exceed 96 columns, an error (ERROR 6) occurs if the 96 column is in the middle of a numeric value.

In format (4), the specified value is displayed from the left side of the display. The column following the end of this displayed value is specified as the display starting position of display commands such as for the next PRINT command.

Note: Do not combine display commands (PRINT, etc.) with serial I/O commands (LPRINT, etc. for the serial I/O interface).
Combining them may clear the display start position specified in format (4).

In format (5), the previously displayed value is displayed as is.

The sixth and seventh forms of the PRINT statement do no print. The sixth form causes all PRINT statements which follow it in the program to be treated as if they were LPRINT statements. The seventh form resets this condition so that the PRINT statements will again work with the display. If it is input manually, an error results.

Examples

```
                                      Display

1Ø A=123:B=5/9:X$="ABCDEF":
   Y$="VWXYZ"                         ABCDEF              5.55555E-Ø1
2Ø PRINT X$,B

3Ø PRINT A;B                          123.5.555555556E-Ø1

4Ø PRINT X$;A;                        ABCDEF123.

5Ø PRINT Y$;B                         ABCDEF123.VWXYZ5.5555555
                                      56E-Ø1
```

```
1 PRINT # "var list"
2 PRINT # "filename" ; var list

Where:  var list       is:  variable
                       or:  variable, var list

Abbreviations:  P. #,  PR. #,  PRI. #,  PRIN. #

See also:  INPUT #,  PRINT,  READ
```

Purpose

The PRINT # verb is used to store values on the cassette tape.

Use and Examples

The following variable types can be used for variable names:
(1) Fixed variables — A, B, X, A(26), C*, A(1Ø)*, etc.
(2) Simple variables — AA, B2, XY$, etc.
(3) Array variables — B(*), CD(*), N$(*), etc.

1) Saving fixed variable contents onto tape

The contents of fixed variables can be saved onto tape by specifying the desired variable names (separated by commas) in the PRINT # statement.

```
PRINT # "DATA 1"; A, B, X, Y
```

This statement saves contents of variables A, B, X, and Y into the tape file named "DATA 1".
If you wish to save the contents of the specified fixed variable and all the subsequent fixed variables, subscript that variable name with an asterisk *.

PRINT # "D-2"; D* — This statement saves the contents of fixed variables D through Z (and of extended variables A(27) and beyond, if defined) into the tape file named "D-2".

PRINT # E, X$, A(3Ø)* — This statement saves the contents of the fixed variables E and X$ and of the extended variables A(3Ø) and all the remaining variables, onto the tape without a filename.

Note: Subscripted fixed variable names A(1) through A(26) can be specified in the PRINT # statement in much the same way as A through Z (or A$ through Z$). However, if array A is already defined by the DIM statement, A( ) cannot be used to define subscripted fixed variables.

2) Saving simple variable (two-character variable) contents

The contents of simple variables can be saved onto tape by specifying the desired variable names.

```
PRINT # "DM-1"; AB, Y1, XY$
```

This statement saves the contents of the simple variables AB, Y1, and XY$ into the tape file named 'DM-1'.

3) Saving array variable contents

The contents of all variables of a specific array can be saved onto tape by specifying the array name subscripted by an asterisk enclosed in parentheses (*).

```
PRINT # "DS-2"; X(*), Y$(*)
```

This statement saves the contents of all the elements (X(Ø), X(1), ...) of the array X, and of all the elements (Y$(Ø), Y$(1), ...) of the array Y$, into the tape file name 'DS-2'.

Note: It is not possible to save the contents of only one or more specific elements of an array. While fixed variables or subscripted fixed variables allow you to save only specific variables, an array (such as A), defined by the DIM statement does not allow you to save only a specific part of it.

* If the PRINT # statement is executed with no variable names specified, an error (ERROR 1) will result.

— CAUTION —

The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be set aside in the program/data area before the PRINT # statement is executed. Otherwise, the execution of the PRINT # statement for undefined variables will result in an error.

1 RADIAN

Abbreviations: RAD., RADI., RADIA.

See also: DEGREE and GRAD

Purpose

The RADIAN verb is used to change the form of angular values to radian form.

Use

The PC-2500 has three forms for representing angular values—decimal degrees, radians, and gradient. These forms are used in specifying the arguments to the SIN, COS, and TAN functions and in returning the results from the ASN, ACS, and ATN functions.

The RADIAN function changes the form for all angular values to radian form until a DEGREE or GRAD verb is used. Radian form represents angles in terms of the length of the arc with respect to a radius, i.e., 36Ø° is 2 PI radians since the circumference of a circle is 2 PI times the radius.

Examples

```
1Ø RADIAN
2Ø X=ASN 1            X now has a value of 1.57Ø796327 or PI/2, the Arcsine of 1.
3Ø PRINT X
```

1 RANDOM

Abbreviations: RA., RAN., RAND., RANDO.

Purpose

The RANDOM verb is used to reset the seed for random number generation.

Use

When random numbers are generated using the RND function, the PC-2500 begins with a predetermined "seed" or starting number. The RANDOM verb resets this seed to a new randomly determined value.

The starting seed will be the same each time the PC-2500 is turned on, so the sequence of random numbers generated with RND is the same each time, unless the seed is changed. This is very convenient during the development of a program because it means that the behavior of the program should be the same each time it is run even though it includes a RND function. When you want the numbers be truly random, the RANDOM statement can be used to make the seed itself random.

Examples

1Ø RANDOM 2Ø X=RND 1Ø	When run from line 2Ø, the value of X is based on the standard seed. When run from line 1Ø, a new seed is used.

1 **READ** variable list

Where: variable list   is: variable
                        or: variable , variable list

Abbreviations: REA.

See also: DATA, RESTORE

Purpose

The READ verb is used to read values from a DATA statement and assign them to variables.

Use

When assigning initial values to an array, it is convenient to list the values in a DATA statement and use a READ statement in a FOR...NEXT loop to load the values into the array. When the first READ is executed, the first value in the first DATA statement is returned. Succeeding READs use succeeding values in the sequential order in which they appear in the program, regardless of how many values are listed in each DATA statement or how many DATA statements are used.

If desired, the values in a DATA statement can be read a second time by using the RESTORE statement.

Examples

```
1Ø DIM B (1Ø)                    Set up an array
2Ø WAIT 32
3Ø FOR H=1 TO 1Ø
4Ø READ B(H)                     Loads the values from the DATA statement into
5Ø PRINT B(H) * 2;                 B(  )—B(1) is 1Ø, B(2) is 2Ø, B(3) is 3Ø, etc.
6Ø NEXT H
7Ø DATA 1Ø, 2Ø, 3Ø, 4Ø, 5Ø, 6Ø
8Ø DATA 7Ø, 8Ø, 9Ø, 1ØØ
9Ø END
```

1 **REM** remark

Abbreviations: none

Purpose

The REM verb is used to include comments in a program.

Use

Often it is useful to include explanatory comments in a program. These can provide titles, names of authors, dates of last modification, usage notes, reminders about algorithms used, etc. These comments are included by means of the REM statement.

The REM statement has no effect on the program execution and can be included anywhere in the program. Everything following the REM verb in that line is treated as a comment.

Examples

```
1Ø REM THIS LINE HAS NO EFFECT
```

1 **RESTORE**
2 **RESTORE** expression

Abbreviations: RES., REST., RESTO., RESTOR.

See also: DATA, READ

Purpose

The RESTORE verb is used to reread values in a DATA statement or to change the order in which these values are read.

Use

In the regular use of the READ verb PC-2500 begins reading with the first value in a DATA statement and proceeds sequentially through the remaining values. The first form of the RESTORE statement resets the pointer to the first value of the first DATA statement, so that it can be read again. The second form of the RESTORE statement resets the pointer to the first value of the first DATA statement whose line number is greater than the value of the expression.

Examples

```
10 DIM B(10)          Sets up an array.
20 WAIT 32
30 FOR H=1 TO 10
40 RESTORE
50 READ B(H)          Assign the value 20 to each of the elements of B(  ).
60 PRINT B(H) * H;
70 NEXT H
80 DATA 20
90 END
```

1 RETURN

Abbreviations: RE., RET., RETU., RETUR.

See also: GOSUB, ON ... GOSUB

Purpose

The RETURN verb is used at the end of a subroutine to return control to the statement following the originating GOSUB.

Use

A subroutine may have more than one RETURN statement, but the first one executed terminates the execution of the subroutine. The next statement executed will be the one following the GOSUB or ON ... GOSUB which calls the subroutine. If a RETURN is executed without a GOSUB, an ERROR 5 will occur.

Examples

```
1Ø GOSUB 1ØØ
2Ø END
1ØØ PRINT "HELLO"
11Ø RETURN
```

When run, this program prints the word "HELLO" once.

1 STOP

Abbreviations: S., ST., STO.,

See also: END; CONT command

Purpose

The STOP verb is used to halt execution of a program for diagnostic purposes.

Use

When the STOP verb is encountered in program execution, the PC-2500 execution halts and a message is displayed such as 'BREAK IN 2ØØ' where 2ØØ is the number of the line containing the STOP. STOP is used during the development of a program to check the flow of the program or examine the state of variables. Execution may be restarted using the CONT command.

Examples

1Ø STOP	Causes "BREAK IN 1Ø" to appear in the display.

1 TROFF

Abbreviations: TROF.

See also: TRON

Purpose

The TROFF verb is used to cancel the trace mode.

Use

Execution of the TROFF verb restores normal execution of the program.

Examples

```
1Ø TRON
2Ø FOR H=1 TO 3
3Ø NEXT H
4Ø TROFF
```

When run, this program displays the line numbers 1Ø, 2Ø, 3Ø, 3Ø, 3Ø and 4Ø as the ⬇ is pressed.

1 TRON

Abbreviations: TR., TRO.

See also: TROFF

Purpose

The TRON verb is used to initiate the trace mode.

Use

The trace mode provides assistance in debugging programs. When the trace mode is on, the line number of each statement is displayed after each statement is executed. The PC-2500 then halts and waits for the Down Arrow key to be pressed before moving on to the next statement. The Up Arrow key may be pressed to see the statement which has just been executed. The trace mode continues until a TROFF verb is executed or the key operation of [SHIFT] + [CLS] is performed.
After a result is displayed at the position specified in the CURSOR command during the trace mode, the next line number is displayed on the next line of the display. (See page 128 on the CURSOR command.)
When, during the trace mode, the display start position is specified after execution of the CURSOR command, it is cleared if variables are called or calculations are made in manual operation.

Examples

```
1Ø TRON
2Ø FOR H=1 TO 3
3Ø NEXT H
4Ø TROFF
```

When run, this program displays the line numbers 1Ø, 2Ø, 3Ø, 3Ø, 3Ø and 4Ø as the [↓] is pressed.

1 **USING**
2 **USING** "editing specification"
3 **USING** character variable

Abbreviations: U., US., USI., USIN.

See also: LPRINT, PAUSE, PRINT

Further guide to the use of USING is provided in Appendix C

Purpose

The USING verb is used to control the format of displayed or printed output.

Use

The USING verb can be used by itself or as a clause within a LPRINT, PAUSE, or PRINT statement. The USING verb establishes a specified format for output which is used for all output which follows until changed by another USING verb.

The editing specification of the USING verb consists of a quoted string composed of some combination of the following editing characters:

# Right justified numeric field character
. Decimal point.
^ Used to indicate that numbers should be displayed in scientific notation.
& Left justified alphanumeric field.

For example, "####" is an editing specification for a right justified numeric field with room for 3 digits and the sign. In numeric fields, a location must be included for the sign, even if it will always be positive.

Editing specifications may include more than one field. For example "####&&&&" could be used to print a numeric and a character field next to each other.

If the editing specification is missing, as in format 1, special formatting is turned off and the built-in display rules pertain.

Examples

	Display
1Ø A=125: X$="ABCDEF"	
2Ø PRINT USING "##.##^";A	1.25E Ø2
3Ø PRINT USING "&&&&&&&&";X$	ABCDEF
4Ø PRINT USING "####&&&";A;X$	125ABC

1 **WAIT** expression
2 **WAIT**

Abbreviations: W., WA., WAI.

See also: PAUSE, PRINT

Purpose

The WAIT verb is used to control the length of time that displayed information is shown before program execution continues.

Use

In normal execution the PC-2500 halts execution after a PRINT, GPRINT, PSET, PRESET, LINE command until the ENTER key is pressed. The WAIT command causes the PC-2500 to display for a specified interval and then proceed automatically (similar to the PAUSE verb). The expression which follows the WAIT verb determines the length of the interval. The interval may be set to any value from Ø to 65535. Each increment is about one fifty-nine of a second. WAIT Ø is too fast to be read reasonably; WAIT 65535 is about 19 minutes. WAIT with no following expression resets the PC-2500 to the original condition of waiting until the ENTER key is pressed.

Examples

1Ø WAIT 59	Causes PRINT to wait about 1 second.

FUNCTIONS

Pseudovariables

1 INKEY$

If a key is pressed while the INKEY$ command is executed, its content is read and assigned to the specified variable.

The INKEY$ command is usually used in a loop (as shown below) to wait until a valid key is pressed.

Example

```
  1Ø CLS : WAIT 6Ø
  2Ø Z$=INKEY$
  3Ø IF Z$="" THEN 2Ø
  4Ø CURSOR 8, 1
  5Ø PRINT "---";Z$;"---"
  6Ø GOTO 1Ø
```

These lines are repeatedly executed until a key is pressed.

- If no key is pressed while the INKEY$ command is executed, a null character (blank) is assigned to the variable.
- The contents of the MODE , PEN , SHIFT + PEN , SHIFT + ↑ , SHIFT + ↓ , and SHIFT + BRK keys are null characters (blank) when read by the INKEY$ command.
  The BRK key (break key) functions to temporarily stop the program execution.
  The CAPS , SHIFT , and DEF keys function according to their respective functions.
- If an INKEY$ command is written at the beginning of the program, the start key may be read (by the INKEY$ command) when the program is started. For example, in the following program

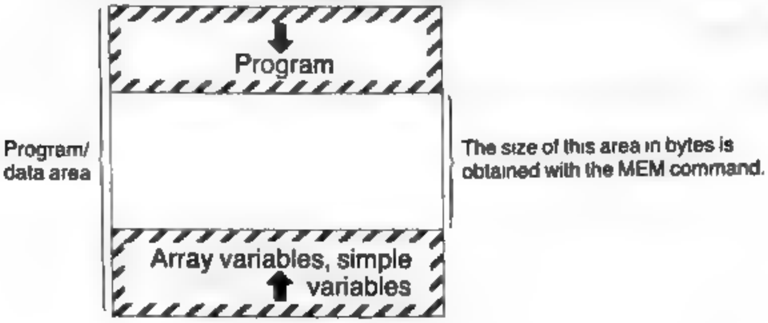
```
    1Ø "Z": Z$=INKEY$
    ⋮
```

  the Z key may be read when the program is started by pressing the DEF Z keys.
- The following codes are obtained after reading the keys with the INKEY$ command and converting the results with the ASC function.

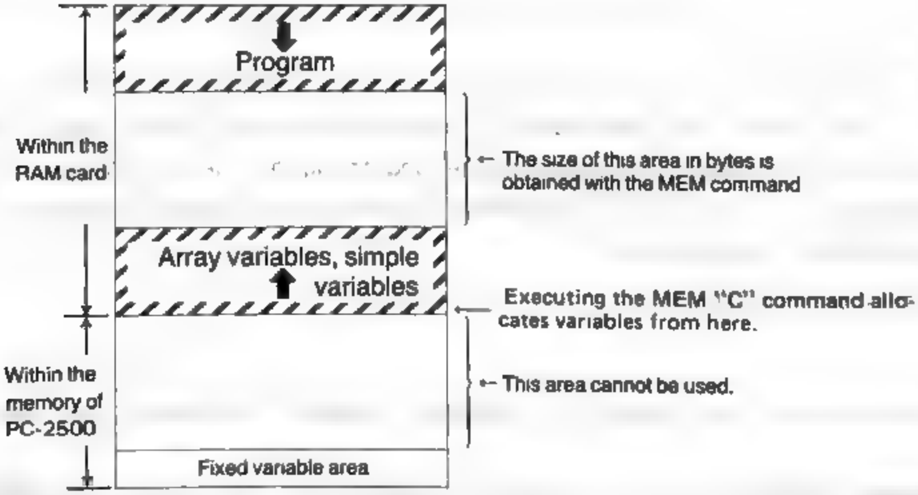
Key	Code	Key	Code	Key	Code
SHIFT + ←	2	ENTER	13	←	29
SHIFT + →	6	INS	18	↑	30
DEL BS	8	SHIFT + CLS	26	↓	31
CLS	12	→	28	SHIFT + DEL BS	127

```
1 MEM
1 MEM "C"
2 MEM "B"
```

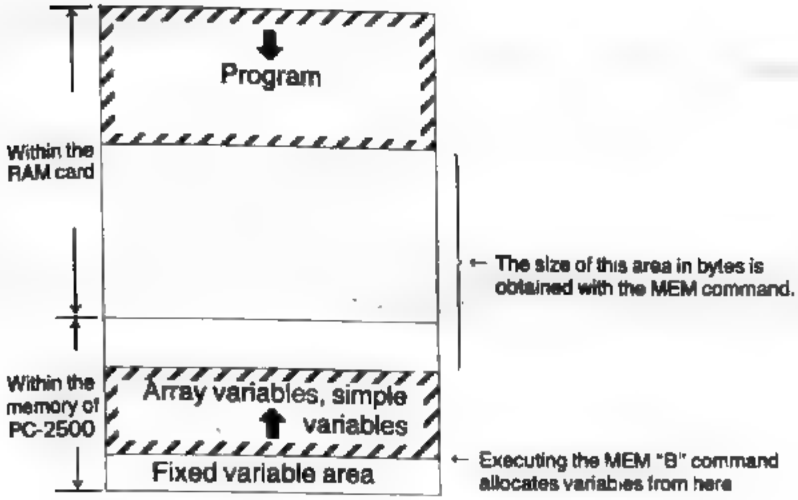
When executing format (1), the number of free bytes (area not used by a program, array variables, or simple variables) in the program/data area is obtained.


Program
Program/ data area
The size of this area in bytes is obtained with the MEM command.
Array variables, simple variables

When the optional RAM card has been mounted, format (2) can be specified so that variables are allocated on the RAM card.


Program
Within the RAM card
The size of this area in bytes is obtained with the MEM command
Array variables, simple variables
Executing the MEM "C" command allocates variables from here.
Within the memory of PC-2500
This area cannot be used.
Fixed variable area

Format (3) restores the start position of the variable allocation area which was set in the memory of the RAM card by format (2) to the memory in the PC-2500.


Program
Within the RAM card
← The size of this area in bytes is obtained with the MEM command.
Within the memory of PC-2500
Array variables, simple variables
Fixed variable area
← Executing the MEM "B" command allocates variables from here

If variables have already been allocated, switching between MEM "C" and MEM "B" will result in the following:

From MEM "B" to MEM "C": Variables within the PC-2500 are transferred to the RAM card.

From MEM "C" to MEM "B": Variables within the RAM card are transferred to the PC-2500.

MEM "B" and MEM "C" can also be switched in "Use RAM Card" for the business program (see page 353).

- If a RAM card is not mounted, execution of MEM "C" results in an error (ERROR 1).
- The setting for the MEM "C" command is stored within the RAM card. Removing the RAM card from the PC-2500 invalidates the setting. (The setting changes to MEM "B".) Installing the RAM card once again restores the setting.

1 MEM$

This command determines whether the current start position of the variable allocation area is in the memory of the RAM card or in the memory of the PC-2500.

[Example]
MEM$ ENTER

If the start position of the variable allocation area is within the PC-2500 (set by MEM "B"), "B" is displayed. If it is within the RAM card (set by MEM "C"), "C" is displayed.

- "B" is displayed when the RAM card is not installed.

```
    1  PI
```

PI is a numeric pseudovariable which has the value of PI. Like other numbers the value of PI is kept to 1Ø digit accuracy (3.141592654).

Numeric Functions

Numeric functions are a group of mathematical operations which take a single numeric value and return a numeric value. They include trigonometric functions, logarithmic functions, and functions which operate on the integer and sign parts of a number. Many dialects of BASIC require that the argument to a function be enclosed in parentheses. The PC-2500 does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument.

LOG 1ØØ+1ØØ will be interpreted as:
(LOG 1ØØ)+1ØØ not LOG (1ØØ+1ØØ).

1 **ABS** numeric expression

ABS is a numeric function which returns the absolute value of the numeric argument. The absolute value is the value of a number without regard to its sign. ABS −1Ø is 1Ø.

1 **ACS** numeric expression

ACS is a numeric function which returns the arccosine of the numeric argument. The arccosine is the angle whose cosine is equal to the expression. The value returned depends on whether the PC-2500 is in decimal degree, radian, or gradient mode for angles. ACS.5 is 6Ø in the decimal degree mode.

1 **ASN** numeric expression

ASN is a numeric function which returns the arcsine of the numeric argument. The arcsine is the angle whose sine is equal to the expression. The value returned depends on whether the PC-2500 is in decimal degree, radian, or gradient mode for angles. ASN.5 is 3Ø in the decimal degree mode.

1 **ATN** numeric expression

ATN is a numeric function which returns the arctangent of the numeric argument. The arctangent is the angle whose tangent is equal to the expression. The value returned depends on whether the PC-2500 is in decimal degree, radian, or gradient mode for angles. ATN 1. is 45 in the decimal degree mode.

1 **COS** numeric expression

COS is a numeric function which returns the cosine of the angle argument. The value returned depends on whether the PC-2500 is in decimal degree, radian, or gradient mode for angles. COS 6Ø is .5 in the decimal degree mode.

1 **DEG** numeric expression

The DEG function converts an angle argument in DMS (Degree, Minute, Second) format to DEG (Decimal Degree) form. In DMS format the integer portion of the number represents the degrees, the first and second digits of the decimal represent the minutes, the third and fourth digits of the decimal represent the seconds, and any further digits represent decimal seconds. For example, 55° 1Ø′ 44.5″ is represented as 55.1Ø445. In DEG format the integer portion is degrees and the decimal portion is decimal degrees. DEG 55.1Ø445 is 55.179Ø2778.

1 **DMS** numeric expression

DMS is a numeric function which converts an angle argument in DEG format to DMS format (see DEG). DMS 55.179Ø2778 is 55.1Ø445.

1 **EXP** numeric expression

EXP is a numeric function which returns the value of e (2.718281828—the base of the natural logarithms) raised to the value of the numeric argument. EXP 1 is 2.718281828.

1 **INT** numeric expression

INT is a numeric function which returns the integer part of its numeric argument. INT PI is 3.

1 **LN** numeric expression

LN is a numeric function which returns the logarithm to the base e (2.718281828) of its numeric argument. LN 1ØØ is 4.6Ø517Ø186.

1 **LOG** numeric expression

LOG is a numeric function which returns the logarithm to the base 1Ø of its numeric argument. LOG 1ØØ is 2.

1 **RND** numeric expression

RND is a numeric function which generates random numbers. If the value of the argument is less than 1 but greater than or equal to zero, the random number is less than 1 and greater than or equal to zero. If the argument is an integer greater than or equal to 1, the result is a random number greater than or equal to 1 and less than or equal to the argument. If the argument is greater than or equal to 1 and not an integer, the result is a random number greater than or equal to 1 and less than or equal to the smallest integer which is larger than the argument: (In this case, the generation of the random number changes depending on the value of the decimal portion of the argument.):

	------------- Result	-------------
Argument	Lower Bound	Upper Bound
.5	Ø<	<1
2	1	2
2.5	1	3

The same sequence of random numbers is normally generated because the same "seed" is used each time the PC-2500 is turned on. To randomize the seed, see the RANDOM verb.

1 **SGN** numeric expression

SGN is a numeric function which returns a value based on the sign of the argument. If the argument is positive, the result is 1; if the argument is zero, the result is Ø; if the argument is negative, the result is −1. SGN −5 is −1.

1 **SIN** numeric expression

SIN is a numeric function which returns the sine of the angle argument. The value returned depends on whether the PC-2500 is in decimal degree, radian, or gradient mode for angles. SIN 3Ø is Ø.5 in the decimal degree mode.

1 SQR numeric expression

SQR is a numeric function which returns the square root of its argument. SQR 4 is 2.

1 TAN numeric expression

TAN is a numeric function which returns the tangent of its angle argument. The value returned depends on whether the PC-2500 is in decimal degree, radian, or gradient mode for angles. TAN 45 is 1 in the decimal degree mode.

String Functions

String functions are a group of operations used for manipulating strings. Some take a string argument and return a numeric value. Some take a string argument and return a string. Some take a numeric value and return a string. Some take a string argument and one or two numeric arguments and return a string. Many dialects of BASIC require the argument of a function to be enclosed in parentheses. The PC-2500 does not require these parentheses, except when it is necessary to indicate what part of a more complex expression is to be included in the argument. String functions with two or three arguments all require the parentheses.

1 ASC string expression

ASC is a string function which returns the numeric character code value of the first character in its argument. The chart of character codes and their relationship to characters is given in Appendix B. ASC "A" is 65.
The PC-2500 uses ASCII codes and their characters.

1 CHR$ numeric expression

CHR$ is a string function which returns the character which corresponds to the numeric character code of its argument. The chart of character codes and their relationship to characters is given in Appendix B. CHR$ 65 is "A".

Note: If character code 13 is specified when manually executing the CHR$ command, the specified contents that follow it will not be displayed.

[Examples]

```
CHR$7Ø+CHR$71+CHR$13+CHR$75+CHR$76
ENTER → FG
```

Characters K and L for codes 75 and 76 are not displayed.

1 LEFT$ (string expression, numeric expression)

LEFT$ is a string function which returns the leftmost part of the string first argument. The number of characters returned is determined by the numeric expression. LEFT$ ("ABCDEF", 2) is "AB".

1 **LEN** string expression

LEN is a string function which returns the length of the string argument. LEN "ABCDEF" is 6.

1 **MID$** (string expression, num. exp. 1, num. exp. 2)

MID$ is a string function which returns a middle portion of the string in the first argument. The first numeric argument indicates the first character position to be included in the result. The second numeric argument indicates the number of characters that are to be included. MID$ ("ABCDEF", 2,3) is "BCD".

1 **RIGHT$** (string expression, numeric expression)

RIGHT$ is a string function which returns the rightmost part of the string first argument. The number of characters returned is determined by the numeric argument. RIGHT$ ("ABCDEF", 3) is 'DEF'.

1 **STR$** numeric expression

STR$ is a string function which returns a string which is the character representation of its numeric argument. It is the reverse of VAL. STR$ 1.59 is '1.59'.

1 **VAL** string expression

VAL is string function which returns the numeric value of its string argument. It is the reverse of STR$. The VAL of a non-number is zero. VAL "1.59" is 1.59.

Note: The character-string convertible by VAL function to a numerical value consists of numerals (Ø to 9.), symbols (+ and –) and a symbol (E) indicating an exponential portion. No other characters and symbols are included. If a character-string includes other characters and symbols, any character-string on the right of that character-string will be ignored. If included in a character-string, a space is usually regarded as non-existing.

GRAPHIC RELATED COMMANDS

1 GCURSOR (expression 1, expression 2)

Abbreviation: GC., GCU., GCUR., GCURS., GCURSO.

See also: CURSOR, GPRINT

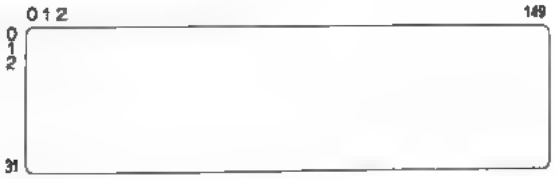
Purpose

Specifies the display start position in dot (point) units.

Use

The GCURSOR command specifies the start position of the display for the GPRINT command. (Moves the graphic cursor to the specified position.)

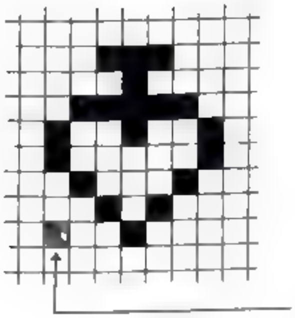
- The display unit is composed of 15Ø horizontal and 32 vertical dots (points). Each dot is assigned a number ranging Ø—149 in the X direction and Ø—31 in the Y direction. The display start position is specified by specifying the number in the X direction with expression 1 in the format above and the number in the Y direction with expression 2.



Example

```
1Ø GCURSOR (71, 2Ø)
2Ø GPRINT "1824458F452418"
```

Executing this program displays the following near the center of the display unit. (The shaded part is not displayed.)



In the program above, this shaded position is the display start position (71, 20).

Note: The values of expression 1 and expression 2 can be specified in the range of −32768 to +32767. However, if the values for expression 1 and expression 2 exceed the ranges of Ø–149 and Ø–31, respectively, the specified position will be outside the boundaries of the display unit and a vertical position (position which does not actually exist) will be specified for the display start position.

* The display start position of (Ø, 7) will be specified when the RUN command or CLS command is executed or when SHIFT + CLS are pressed.
When the program is started using the GOTO command or defined key, the value for the y-direction is retained. The x-direction returns to Ø.

1 **GPRINT** string
2 **GPRINT** expression; expression; expression; ...
3 **GPRINT**

Abbreviation: GP., GPR., GPRI., GPRIN.

See also: GCURSOR, PRINT

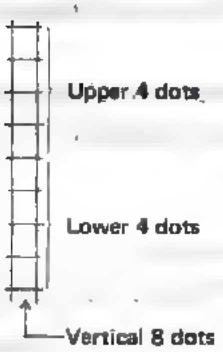
Purpose

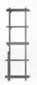
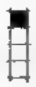
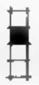
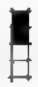
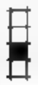
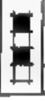
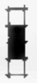
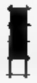
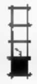
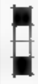
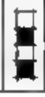
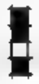
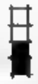
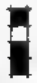
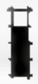
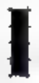
Displays the specified dot pattern.

Use

The GPRINT command displays the specified dot pattern. A vertical line of 8 dots is specified as one dot pattern.

- In format (1), the 8-dot pattern, divided into a lower 4 dots and an upper 4 dots, is specified by a string enclosed within "   ," where the 4-dot patterns are represented by hexadecimal numbers.



Hexadecimal character	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Pattern																

GPRINT "○○ ○○ ○○ ..."

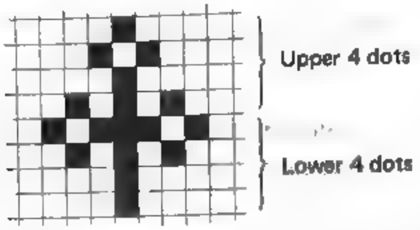
A single group (pair) of 2 numerals specifies one vertical (8-dot) pattern. Any numeral remaining at the end of the string which is not in a pair will be ignored.

Example

GPRINT "1Ø2812FD12281Ø"

Represents the upper (4-dot) pattern.

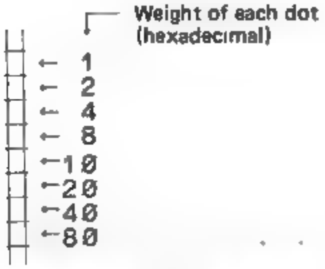
Represents the lower (4-dot) pattern.

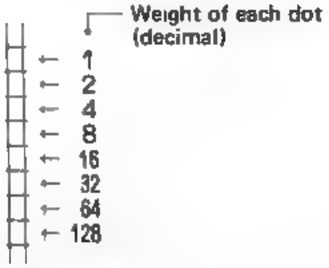


Ø82D28Ø ←— Upper (4-dot) pattern represented in hexadecimal.

121F121 ←— Lower (4-dot) pattern represented in hexadecimal.

* In format (2), a vertical 8-dot pattern is specified as a hexadecimal or decimal value. A "weight" is assigned to each dot in the vertical 8-dot pattern as shown below.



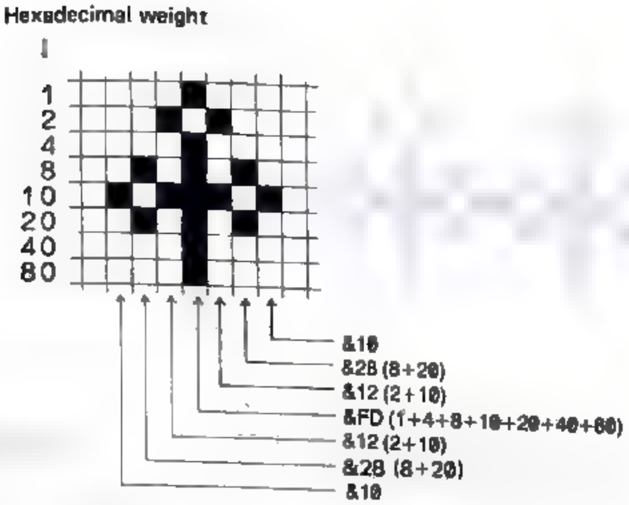


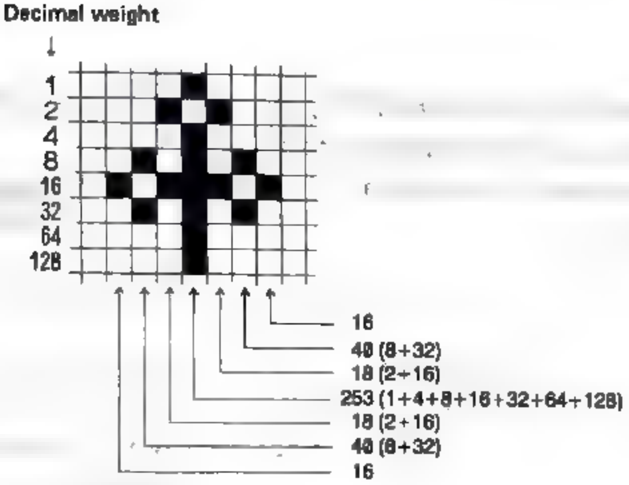
Specify the dot pattern with a numeric value with the sum of the weights of the dots to be lit on the display unit.

Example

* Specifying dot patterns in hexadecimal.
GPRINT &1Ø; &28; &12; &FD; &12; &28; &1Ø

* Specifying dot patterns in decimal.
GPRINT 16; 4Ø; 18; 253; 18; 4Ø; 16



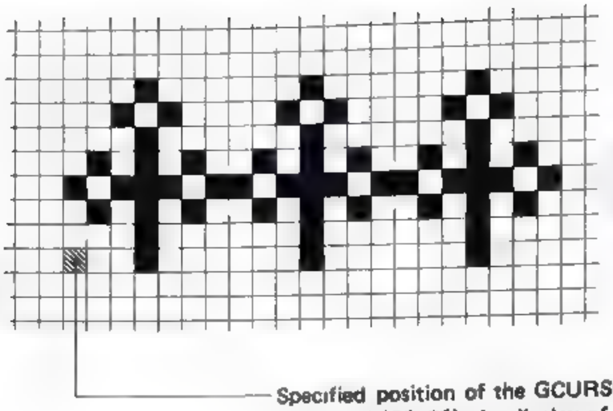


In format (3), the previous graphic display continues to be displayed.

* When the display start position has been specified with the GCURSOR command, the first dot pattern (out of the entire pattern displayed with the GPRINT command) is displayed using the upper 8 dots which includes the specified display start position (dot).

Example

```
1Ø AA$="1Ø2812FD12281Ø"
2Ø GCURSOR (6Ø, 2Ø)
3Ø GPRINT AA$;AA$;AA$
```



Specified position of the GCURSOR command (6Ø, 2Ø), the display of the pattern starts from this position using the upper 8 dots.

Note: When the GPRINT statement ends with a ";", the column to the right of the display after execution becomes the display start position. When the GPRINT statement ends with a ":" or ENTER , the value for the x-direction returns to Ø.

```
                     (A)                                  (B)                    ⎧S⎫
1 LINE  (expression 1, expression 2)—(expression 3, expression 4),  ⎨R⎬ ,
                                                                                 ⎩X⎭
   expression 5, ⎧B ⎫                                                            (C)
                 ⎩BF⎭
        (D)       (E)
```

Abbreviation: LIN.

See also: GCURSOR, PSET

Purpose

Draws a line between 2 specified points.

Use

A line is drawn between the 2 points specified by (expression 1, expression 2) and (expression 3, expression 4).

Example

LINE (Ø, Ø)–(149, 31)

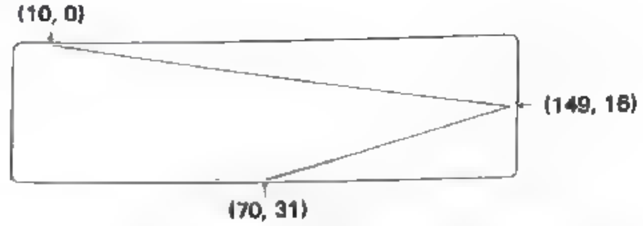
A line from the top left to the bottom right of the display (screen) is drawn.

* The values of expression 1—expression 4 in terms (A) and (B) can be specified within the range of –32768 to +32767, though to be specified within the screen, expression 1 and expression 3 must be within the range of Ø–149, and expression 2 and expression 4, the range of Ø–31.
Even if a point outside the boundaries of the screen is specified but is within the range of –32768 to +32767, an error does not result. Only the portions corresponding to the area within the boundaries of the screen are displayed. Specifying a point beyond the range results in an error (ERROR 3).

* Term (A) (expression 1, expression 2) can be omitted. If omitted, the line is drawn from either position (Ø, Ø) or the position specified by term (B) (expression 3, expression 4) in a LINE command executed directly before.

Example

```
 5 CLS : WAIT Ø
1Ø LINE (1Ø, Ø)–(149, 16)
2Ø WAIT : LINE–(7Ø, 31)
```



Note: Since the screen is composed of dots, diagonal lines may not appear to be straight. Further, curves may not be accurately displayed.

* S, R, or X in term (C) specify the dots on the line drawn to light up, clear or reverse.

S: Draws the line with the dots lighted. (Sets the dots.)

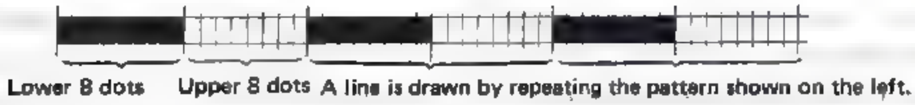
R: Draws the line with the dots cleared. This is used to draw a line in an area where the surrounding dots are lighted or to clear an existing line. (Resets the dots.)

X: Draws the line and clears the dots if already lit or lights the dots if not lit. (Reverses the dots.)

If none of S, R, or X is specified, S is assumed.

* The value of expression 5 in term (D) specifies the type of line.

For example, when the value of expression 5 is 255 (&ØØFF), the following type of line will be drawn.



The number 255 (&ØØFF) can be expressed as a binary number as follows:

ØØØØØØØØ11111111

If the 16 dots of the line shown in the figure above and the binary number are compared, it can be seen that the dots corresponding to the 1's are lit and the dots corresponding to the Ø's are cleared. In this manner, the type of line is specified by the Ø's and 1's after converting the value of expression 5 into a 16 bit binary number. Therefore, the line does not appear on the screen when the value of expression 5 is Ø and a solid line appears when the value is 65535 (&FFFF). A solid line is also displayed if expression 5 is omitted. However, if R is specified in term (C), the opposite occurs, and if X is specified, the dots corresponding to the 1's are reversed.

* The value of expression 5 can be specified in the range of Ø-65535 (&FFFF).

* Term (E) draws a square, the diagonal of which is a line connecting 2 points specified by terms (A) and (B).

B: Draws a square.

BF: Draws a square filled in with lines.

Examples

```
10 CLS : WAIT 0
20 AA$="102812FD122810"
30 GCURSOR (64, 20)
40 GPRINT  AA$; AA$; AA$
50 LINE (24, 0)-(124, 31), &F18F, B
60 LINE (34, 3)-(114, 28), X, BF
70 GOTO  60
```

1 POINT (expression 1, expression 2)

Abbreviation: POI., POIN.

See also: GCURSOR, PSET, PRESET

Purpose

Reads the state of the specified dot.

Use

If the dot specified by (expression 1, expression 2) is lit, a "1" is returned, and if cleared, a "Ø" is returned.

If the specified dot lies beyond the boundaries of the screen, a "−1" is returned.

* The values of expression 1 and expression 2 can be specified within the range of −32768 to +32767. However, dots on the screen are in the range of Ø–149 for expression 1 and Ø–31 for expresion 2.

Example

```
 1Ø CLS : WAIT Ø: A=75
 2Ø LINE (5Ø, Ø)−(5Ø, 31)    ←┐ Draws 2 vertical lines.
 3Ø LINE (1ØØ, Ø)−(1ØØ, 31)  ←┘
 4Ø PSET (A, 16)             ←  Lights up a dot (point) between the 2 lines.
 5Ø B= POINT (A+1, 16)       ←  Checks to see whether the next dot on the right
                                is lit.
 6Ø IF B THEN 15Ø            ←  If lit, jump to line 15Ø.
 7Ø PSET (A+1, 16)           ←  If cleared, light it.
 8Ø PRESET (A, 16)           ←  Then, clear the dot lit up earlier.
 9Ø A=A+1                    ←  Move one dot to the right.
1ØØ GOTO 5Ø                  ←  Go back to line 5Ø.
15Ø B=POINT (A−1, 16)        ←  Checks to see whether the dot to the left of the
                                lit dot is lit.
16Ø IF B THEN 5Ø             ←  If lit, go to line 5Ø.
17Ø PSET (A−1, 16)           ←  If cleared, light it.
18Ø PRESET (A, 16)           ←  Then, clear the dot lit up earlier.
19Ø A=A−1                    ←  Move one dot to the left.
2ØØ GOTO 15Ø                 ←  Go back to line 15Ø.
```

Executing this program moves a dot back and forth between 2 vertical lines drawn on the screen.

1 PRESET (expression 1, expression 2)

Abbreviation: PRE., PRES., PRESE.

See also: PSET, GCURSOR, POINT

Purpose

Clears (resets) the specified dot on the screen.

Use

Clears the dot specified by (expression 1, expression 2).

* The values of expression 1 and expression 2 can be specified within the range of –32768 to +32767. However, dots on the screen are in the range of Ø–149 for expression 1 and Ø–31 for expression 2.

Example

```
1Ø CLS : WAIT Ø
2Ø LINE (2Ø, Ø)–(13Ø, 31), BF
3Ø FOR X=–25 TO 25 STEP Ø.5
4Ø Y=–1*SQR ABS (25*25–X*X)
5Ø PRESET (X+75, Y+31)
6Ø NEXT X
7Ø WAIT : GPRINT
```

Executing this program draws a semicircle inside a filled square.

1 **PSET** (expression 1, expression 2)
2 **PSET** (expression 1, expression 2), X

Abbreviation: PS., PSE.

See also: PRESET, GCURSOR, POINT

Purpose

Lights up or reverses the specified dot on the screen.

Use

- Format (1) lights up the dot specified by (expression 1, expression 2).
- Format (2) clears the dot specified by (expression 1, expression 2) if lit and lights it if cleared.
- The values of expression 1 and expression 2 can be specified within the range of −32768 to +32767. However, dots on the screen are in the range of Ø–149 for expression 1 and Ø–31 for expression 2.

Example

```
1Ø CLS : WAIT Ø : DEGREE
2Ø FOR A=Ø TO 6ØØ
3Ø B=−1* SIN A
4Ø Y= INT (B*16) +16
5Ø X= INT (A/4)
6Ø PSET (X, Y)
7Ø NEXT A
8Ø WAIT : GPRINT
```

Executing this program draws a sine curve on the screen.

SERIAL I/O RELATED COMMAND

1 CLOSE

Abbreviation: CLOS.

See also: OPEN

Purpose

Closes the circuit of the serial I/O interface.

Use

This command closes the circuit (in the software sense) of the serial I/O interface which was opened by the OPEN command.
Therefore, after this command is executed, any output to the serial I/O terminal or input from the same terminal cannot be performed.

1 CONSOLE expression

Abbreviation: CONS., CONSO., CONSOL.

See also: OPEN, LPRINT, LLIST

Purpose

Sets the number of columns per line for data sending.

Use

This command sets the number of columns per line for the LPRINT and LLIST commands when printing characters on the built-in printer or when sending data through the serial I/O interface (terminal).
The PC-2500 sends an end code (CR, LF, or CR+LF) after sending the preset line of data.

- Valid values of the expression are integer in the range of 1–160. If the value of the expression exceeds 160, 160 columns per line will be set. An error (ERROR 3) results if the value is 0 or negative.
  If an expression is not specified, the command is ignored and the number of columns previously set is retained.
- The printing line is set to 39 columns after entering BASIC or after pressing the reset switch.

Note: Use the CONSOLE command in the form CONSOLE expression: LPRINT when printing is to be executed on the built-in printer.

1 INPUT#1 variable, variable, variable...

Abbreviation: I.#1, IN.#1, INP.#1, INPU.#1

See also: OPEN, LPRINT#1

Purpose

Assigns data, input through the serial I/O interface (terminal), to the specified variables.

Use

* This command is valid only when the circuit of the serial I/O interface is open (after the OPEN command is executed) and is ignored otherwise.

* The INPUT#1 command assigns data (sent in the form as described in the section on the PRINT#1 command) to the specified variables.
Therefore, the variables are specified as in the PRINT#1 command.

Example

```
INPUT#1A, AB, C$, E(*)
```

Data input through the I/O interface is assigned to variables A, AB, and C$, and array variable E( ).

* Be sure that the type of both the specified variables and the input data match (i.e. character or numeric types).
In the ASCII code system, if a character is assigned to a numeric variable, its value becomes Ø. If a number is assigned to a character variable, its contents become a character string. Therefore, if the type of both the specified variable and the input data do not match, unexpected values may result.

Even if data in a form such as the function "SIN 3Ø" is given to a numeric variable, it is assumed to be a character string with its content Ø.
For data in the form of "1Ø+4Ø", the characters (numbers) after the operator symbol are ignored. Therefore, the data in this case is "1Ø".

Note 1: If CR (control code: ØDH) or NULL (ØØH) is included within the input data, all data following it may be ignored.

Note 2: Simple variables and array variables must be allocated in the program/data area before executing an INPUT#1 command. An error will result if these variables are not allocated.

1 LLIST

2 LLIST { expression / "label" }

3 LLIST expression 1, expression 2

Abbreviation: LL., LLI., LLIS.

See also: OPEN, CONSOLE

Purpose

Sends the program contents out of the serial I/O interface (terminal).

Use

The LLIST command is valid under manual operation in the PRO or RUN mode. When the circuit of the serial I/O interface is open due to the OPEN command, the program is sent out in ASCII code.
When the circuit is closed, the program is printed on the printer. (See page 73.)

* In format (1), all programs in the PC-2500 are sent out.

[Example]
When the program below is in the PC-2500, pressing

LLIST ENTER

sends out the program in the form shown below.

```
  1Ø: OPEN
1ØØ: REM  **ABC–12**
65279: END
```

Space	1	Ø	:	O	P	E	N	Space	CR
1	Ø	Ø	:	R	E	M	Space	Space	*
*	A	B	C	–	1	2	*	*	CR
6	5	2	7	9	:	E	N	D	Space
CR									

Note: CR is an end code. It is either LF or CR + LF depending on the setting of the OPEN command.

* In format (2), the line indicated by the value of the expression or the line with the specified label is sent out.

- In format (3), the program, from the line indicated by the value of expression 1 to the line indicated by the value of expression 2, is sent out. (Labels can also be used for expression 1 and expression 2.)
  Expression 1 or expression 2 can be omitted in format (3).
- If expression 1 is omitted, the program, from the first line to the line indicated by the value of expression 2, is sent out.
- If expression 2 is omitted, the program, from the line indicated by the value of expression 1 to the last line, is sent out.

- If a line corresponding to the value of expression, expression 1 or expression 2 does not exist, the line with the next largest number which does exist will be specified.
  An error results (ERROR 1) if the lines specified in expression 1 and expression 2 are the same.

- The LLIST command is ignored if a password has been set.

- If programs have been merged using the MERGE command, the LLIST command functions only for the last merged program.
  To list the previously stored programs, execute

```
LLIST "label",
```

- The number of print columns per line is set by the CONSOLE command. If set to 23 columns or less, executing the LLIST command results in an error (ERROR 3).

1 LOAD

Abbreviation: LOA.

See also: OPEN, CLOAD

Purpose

Loads the data sent from the serial I/O interface (terminal) into the program/data area.

Use

The LOAD command is valid when the circuit of the serial I/O interface is open due to the OPEN command. It is ignored when the circuit is closed.

* Data through the serial I/O interface is read until the end code is reached. Data until the end code is considered to be the first line of the program. The PC-2500 converts the data into a form which can be stored as a program and then transfers (writes) it to the program/data area. Then, data is again read from the serial I/O interface, converted in the same way, and then written into the memory.
This operation continues until the text end code (see OPEN command) is read.

* Up to 256 bytes of data can be read at a time. Therefore, if more than 256 bytes of data are sent before the end code is read, an error results.

* The data which has been read is converted and then written to the program/data area. If one line, including the line number, exceeds 80 bytes, an error results. Further, an error also results if the beginning of the line is not a numeric value (line number).

* During execution of the LOAD command, the order of the lines is not rearranged (e.g. ascending order of the line numbers).

Notes:

- Execution of the LOAD command ends when the text end code is read (from the sending side).
  Even if the sending side sends out the entire program, the PC-2500 does not end execution as long as the text end code is not read. In this case, end the execution as follows:
  (1) After the sending side sends the program, have it also send only the text end code.
  (2) Or, press the ON/BRK key to end execution.
- The reserved contents cannot be read from the serial I/O interface.

```
1 LPRINT { expression       }
         { character string }

2 LPRINT { expression       } , { expression       } ... , { expression       }
         { character string }   { character string }       { character string }

3 LPRINT { expression       } ; { expression       } ... ; { expression       }
         { character string }   { character string }       { character string }

4 LPRINT   ... { expression       } ;
               { character string }
               (Format where a ";" is added to the end of 1 and 3 above.)

5 LPRINT
```

Abbreviation: LP., LPR., LPRI., LPRIN.

See also: OPEN, CONSOLE, USING

Purpose

Sends the specified information out through the serial I/O interface (terminal).

Use

When the circuit of the serial I/O interface is opened by the OPEN command, the specified information is sent out through the serial I/O terminal in ASCII code. When the circuit is closed, LPRINT prints information on the printer. (See page 148.)

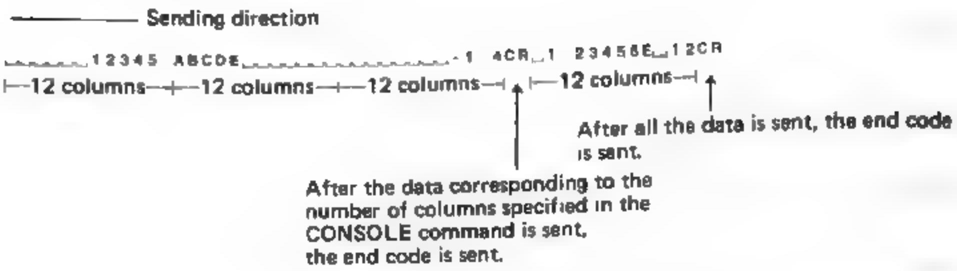
* In format (1), the value of the expression or the character string is sent from its beginning.
  If the value of the expression is negative a "–" sign is sent before the value. If positive, a space is sent.

* In format (2), 12 column (digit) divisions are automatically set. The value of a single expression or a character string is sent within a range of 12 columns (digits).

[Example]

```
1Ø OPEN "12ØØ, N, 8, 1, A, C"
2Ø CONSOLE 36
3Ø LPRINT 12345, "ABCDE",
   –7/5, 1. 23456789E12
```

Executing this program sends information in the following form.


Sending direction
12345 ABCDE -1 4CR 1 23456E 12CR
12 columns
12 columns
12 columns
12 columns
After all the data is sent, the end code is sent.
After the data corresponding to the number of columns specified in the CONSOLE command is sent, the end code is sent.

If the specified character string exceeds 12 columns in this format, only the first 12 characters are sent.
Also, if the value of the expression exceeds 12 digits (in exponential display), it is sent after the low order digits of the fractional part are truncated.
If the value of the expression is negative, a "–" signal is sent before the value. If positive, a space is sent.

- * In format (3), the specified values or strings are sent in the specified sequence. In this format, a space is not sent before a positive number.

  [Example]

  ```
   :
  5Ø LPRINT –123; "ABC"; 567. 89
  ```

  ← Sending direction

  ```
  –123. ABC567. 89CR
  ```

  ↑ End code (LF or CR + LF is sent depending on how the OPEN command is specified.)

- * In format (4), the end code which indicates the end of the data is not sent. However, after the data corresponding to the number of columns specified in the CONSOLE command is sent, the end code is sent.
- * In format (5), only the end code is sent.
- * When the format has been specified in the USING command, formats (1)–(4) send data accordingly.
- * Executing PRINT=LPRINT causes the PRINT command to function as LPRINT command.

Note: To send characters or control codes which cannot be directly entered through the keyboard, specify them using the CHR$ command as shown below.

[Example] To send { }

```
①⋮

  5Ø LPRINT CHR$&7B, CHR$&7D
②⋮

  5Ø A$=CHR$&7B: B$=CHR$&7D
  6Ø LPRINT A$, B$
```

NULL (ØØH) is valid only in (1) and will be ignored in (2).

1 **OPEN** "baud rate, parity, word length, stop bit, type of code, end code, text end code"
2 **OPEN**

Abbreviation: OP., OPE.

See also: CLOSE

Purpose

Allows data to be transferred through the I/O interface. Also sets the I/O conditions.

Use

Format (1) enables data to be transferred through the I/O interface (serial I/O terminal). It also sets the conditions for the data transfer with the connected equipment. The conditions are specified in the following form:
"baut rate, parity, word length, stop bit, type of code, end code, text end code"

Baud Rate: 3ØØ, 6ØØ, 12ØØ
Specifies the modulation rate (transfer rate). For the PC-2500, 3ØØ baud, 6ØØ baud, or 12ØØ baud can be selected.
(1 baud = 1 bit/sec)

Parity: N, E, O
Specifies the type of parity by a character.
N: No parity bit is transmitted nor received.
E: Specifies even parity.
O: Specifies odd parity.

Word Length: 7, 8
Specifies how many bits to be transmitted or received per character. Either 7 or 8 bits can be specified.

Number of Stop Bits: 1, 2

Type of Code: A
Only ASCII codes can be transmitted or received. Therefore, A is always specified.

End Code: C, F, L
Specifies the type of end code to indicate the end of data (delimiting), end of a program line, etc.
C: Specifies the CR (carriage return) code.
F: Specifies the LF (line feed) code.
L: Specifies the CR code + LF code.

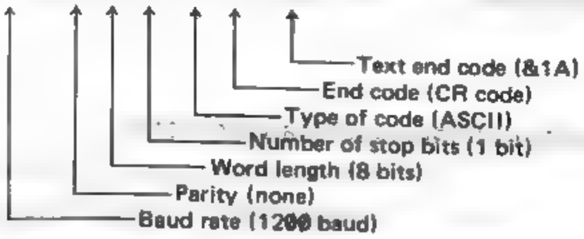
Text End Code: &ØØ–&FF

Specifies the text end code to indicate the end of the program, etc.

(May be required when using the SAVE or LOAD commands.)

Example

```
OPEN "12ØØ, N, 8, 1, A, C, &1A"
```



The conditions in the example above are set after the batteries are replaced or after the reset switch is pressed.

* Any condition specified in the OPEN command can be omitted. If omitted, the current condition remains unchanged.

Example

```
OPEN ",,,2"
```

Only the number of stop bits is changed.

* In format (2), all conditions set previously are retained. This format enables data to be transferred through the I/O interface.

* Executing the OPEN command while the circuit of the I/O interface is open and ready for data transfer (due to prior execution of the OPEN command) results in an error (ERROR 8).

Execute the CLOSE command to close the circuit. (The circuit also closes when the RUN command is executed, when the program ends, or when the power is switch off.)

The previously set conditions are retained even after the CLOSE command is executed.

1 OPEN$

Abbreviation: OP.$, OPE.$

See also: OPEN

Purpose

Obtains the currently set I/O conditions.

Use

The currently set I/O conditions are obtained as a character string.

Example

OPEN$ ENTER 1200, N, 8, 1, A, C, &1A

1 **PRINT#1** variable, variable, variable ...

Abbreviation: P.#1, PR.#1, PRI.#1, PRIN.#1

See also: OPEN, INPUT#1

Purpose

Sends the contents of the specified variables through the serial I/O interface (terminal).

Use

This command is valid only when the circuit of the serial I/O interface is open (due to the OPEN command). It is ignored otherwise.

* Variables are specified as follows.

Fixed Variables: Specify each variable name.
[Example] A, B, C$
Note: Fixed variables cannot be specified in the form of A*

Simple Variables: Specify each variable name.
[Example] AA, B1$, C2

Array Variables: Specify in the form of array name(*).
[Example] B(*), C$(*)
Specified in this manner, the contents of all elements in the array are sent. (Array elements cannot be specified individually).
[Example] 50 PRINT #1A, AB, C$, E(*)

* When data is sent, the end code is added to the end of the contents of each variable. The end code is added to the end of the contents of each element for array variables also.

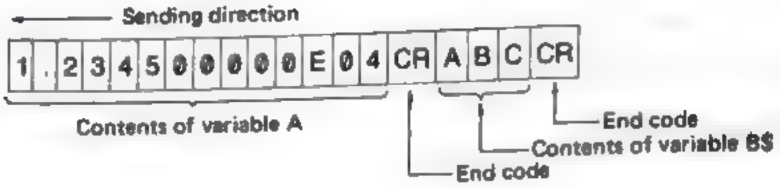
[Example] When

A = 12345 and B$ = "ABC"

executing

PRINT #1A, B$

Sends A and B$ in the following form. (Provided the end code is CR.)



- If the value of the numeric variable is negative, a "–" sign is sent before the value.

* The elements of an array are sent in the following sequence.

1 dimensional array

[Example] For B (3)

B (Ø) → B (1) → B (2) → B (3)

2 dimensional array

[Example] For C (2, 3)

C (Ø, Ø) → C (Ø, 1) → C (Ø, 2) ....

Notes:

* The locations for extended variables such as A(27) and beyond, simple variables, and/or array variables must be allocated in the program/data area before the PRINT #1 command is executed. An error results if an attempt is made to send the contents of a variable which has not been allocated.

An error also results if the type of the specified variable (numeric or character) and that of the variable within the PC-2500 do not match.

1 SAVE

Abbreviation: SA., SAV.

See also: OPEN, LLIST

Purpose

Sends the program within the PC-2500 out through the serial I/O interface (terminal).

Use

When the circuit of the serial I/O interface is open due to the OPEN command, the program is sent in ASCII code.
The command is ignored if the circuit is closed.

* After the entire program is sent, the text end code is sent.

  Note: If the SAVE command is executed while there are several programs stored in the PC-2500 (after using the MERGE command), only the program loaded last will be sent.

* The SAVE command is ignored if a password has been set.

1 BASIC

Abbreviation: BA., BAS., BASI.

See also: TEXT

Purpose

Clears the text mode.
(valid only in manual operation in the program mode)

Use

Executing this command clears the text mode and returns the mode to BASIC. As the mode return to BASIC, the prompt symbol changes from "<" to ">".

- Changing from the text mode to the BASIC mode usually changes the text in the PC-2500 to a program (internal code).
  However, abbreviations such as "P." and "I." are not converted to their respective commands. (In this case, call the cursor to the line and press the ENTER key to convert it to a command.)
  Due to characteristics of the text function, commands and formats included in the text but not found in the PC-2500 may not be executable.
- During program conversion, the "*" mark is displayed on the right end of the 4th line of the display unit.
- If a password has been set, executing the BASIC command results in an error (ERROR 1).

1 TEXT

Abbreviation: TE., TEX.

See also: BASIC

Purpose

Sets the test mode.
(valid only in manual operation in the program mode)

Use

The text function is used when inputting a program written for a higher level personal computer. The program input by the PC-2500 is sent to the personal computer through the serial I/O interface.

- Executing the TEXT command sets the text mode. In the text mode, a number corresponding to the line number and then information corresponding to program commands or data are entered. Then the ENTER key is pressed to write the input to the program/data area.
  However, the written contents, unlike in the BASIC mode, are not converted to commands (internal codes). The text is stored as they are (as characters and/or numbers) in ASCII codes. The text is arranged in the order of the numbers corresponding to the line number at the beginning of each line. (Line number editing function.)
- The text written in the text mode is stored as it is. Therefore, command abbreviations in BASIC (such as I. for INPUT) are displayed and stored as they are.
- If a program is stored in the internal code of the PC-2500 with the text mode set, it is converted to ASCII code.
- During program conversion, the "*" mark is displayed on the right end of the 4th line of the display unit.
- The prompt symbol is "<" in the text mode. (It is usually ">".)
- One line in the text mode (including line number and ENTER) must not exceed 8Ø characters (8Ø bytes). If a line exceeds 8Ø characters due to program conversion, the excess part will be cleared.
- Lower case letters are processed as are in the TEXT mode, and converted into upper case letter in the BASIC mode. (except for those between quotation marks)

80 bytes (end is ENTER)

```
10: PRINT "ABC..... 1234567890"
```

Converted to ASCII code.

```
10: PRINT   "ABC...... 123456
```

80 characters (end is ENTER)

In this example, the PRINT command is one byte in internal code but takes up 6 bytes in ASCII code. Because of this, the last few characters are deleted (7890").

- The number of bytes increases when converting a program from internal code to ASCII code, as shown in the example. If, as a result, the capacity of the program area is exceeded, the program converted up to that point is converted back to internal code and an error will result (ERROR 6).
- If a password has been set, an error (ERROR 1) occurs when the TEXT command is executed.

CHAPTER 10 TROUBLESHOOTING

This chapter provides you with some hints on what to do when your SHARP PC-2500 does not do what you expect it to do. It is divided into two parts—the first part deals with general machine operation and the second with BASIC programming. For each problem there are a series of suggestions provided. You should try each of these, one at a time, until you have fixed the problem.

Machine Operation

If:	Then You Should:
You turn on the machine but there is nothing on the display	1. Check to see that the slide power switch is set to ON position. 2. Push the ON/BRK key to see if AUTO POWER OFF has been activated. 3. Charge the batteries. 4. Adjust the contrast control.
There is a display, but no response to keystrokes	1. Press CLS key to clear. 2. Press SHIFT + CLS to clear. 3. Turn OFF and ON again. 4. Hold down the space bar and push RESET. 5. Push RESET without any key.
You have typed in a calculation or answer and get no response	1. Push ENTER .
You are running a BASIC program and it displays something, and stops	1. Push ENTER .
You enter a calculation and it is displayed in BASIC statement format (colon after the first number)	1. Switch from the PROgram into the RUN mode for calculations.
You get no response from any keys.	1. Hold down the space bar and push RESET. 2. If you get no response from any key even when the above operation is performed, push the RESET without pushing any key. This will clear the program, data and all reserved contents.

BASIC Debugging

When entering a new BASIC program, it is usual for it not to work the first time. Even if you are simply keying in a program that you know is correct, such as those provided in this manual, it is usual to make at least one typing error. If it is a new program of any length, it will probably contain at least one logic error as well. Following are some general hints on how to find and correct your errors.

You run your program and get an error message:

1. Go back to the PROgram mode and use the ↓ or the ↑ keys to recall the line with the error. The cursor will be positioned at the place in the line where the PC-2500 got confused.

2. If you can't find an obvious error in the way in which the line is written, the problem may lie with the values which are being used. For example, CHR$(A) will produce a space if A has a value of 1. Check the values of the variables in either the RUN or the PROgram mode by typing in the name of the variable followed by ENTER .

You RUN the program and don't get an error message, but it doesn't do what you expect.

3. Check through the program line by line using LIST and the ↓ and ↑ keys to see if you have entered the program correctly. It is surprising how may errors can be fixed by just taking another look at the program.

4. Think about each line as you go through the programs as if you were the computer. Take sample values and try to apply the operation in each line to see if you get the result that you expected.

5. Insert one or more extra PRINT statements in your program to display key values and key locations. Use these to isolate the parts of the program that are working correctly and the location of the error. This approach is also useful for determining which parts of a program have been executed. You can also use STOP to temporarily halt execution at critical points so that several variables can be examined.

6. Use TRON and TROFF, either as commands or directly within the program to trace the flow of the program through individual lines. Stop to examine the contents of critical variables at crucial points. This is a very slow way to find a problem, but sometimes it is also the only way.

To continue the program, press the [↓] (Down Arrow) key once. This causes the next line to be executed and its line number to be displayed. Again, you may review the line with the Up Arrow key. You may also check the contents of any variable by typing its name and pressing [ENTER] :

[A] [ENTER] (when A=4 is input before [A] [ENTER] operation)

```
|                                    4|
```

It is necessary to press the [↓] (Down Arrow) key once for each line to be executed until the program ends. If you do not wish to continue normal line-by-line execution, press the [ENTER] key to suspend execution of the program. If you change your mind again, suspended programs may be continued with the CONT command.

A sample session, using our hypotenuse program, follows:

Input	Display
	>
T R O N	TRON_
ENTER	>
R U N	RUN_
ENTER	?
3	3_
ENTER	?
4	4_
ENTER	10:
↑	10: INPUT A, █
↓	20:
↑	20: A=A*A : B=B*█
A	A_
ENTER	9.
B	B_
ENTER	16.
↓	30:
H	H_
ENTER	5.
↓	HYPOTENUSE=5.
↑	40: PRINT "HYPOTENUSE="; █
↓	40:
↓	>

No matter how careful you are, eventually you will create a program which does not do quite what you expect it to. In order to isolate the problem, Sharp's designers have provided a special method of executing programs know as the "Trace" mode. In the Trace mode, the PC-2500 will display the line number of each program line and will halt after the execution of that line. This allows you to follow (or trace) the sequence of instructions as they are actually performed. When the program pauses after the execution of a line, you may inspect or alter the values of variables.

The form of the instruction for initiating the Trace mode is simply: TRON. The TRON instruction may be issued as a command (in RUN mode) or it may be embedded, as a statement, within a program. Used as a command, TRON informs the PC-2500 that tracing is required during the execution of all subsequent programs. The programs to be traced are then started in a normal manner, with a GOTO or RUN command.

If TRON is used as a statement, it will initiate the Trace mode only when the line containing it is executed. If, for some reason, that line is never reached, the Trace mode will remain inactive.

Once initiated, the Trace mode of operation remains in effect until cancelled by a TROFF instruction. The TROFF instruction may also be issued as either a command or a statement. The Trace mode can also be cancelled by the key sequence:

SHIFT + CLS

As an example in using the Trace mode, enter the following program to compute the length of the hypotenuse of a triangle given the length of the sides:

Program Listing:

```
1Ø INPUT A, B
2Ø A=A*A : B=B*B
3Ø H=SQR(A+B)
4Ø PRINT "HYPOTENUSE= ";H
```

In RUN mode, issue the TRON command, followed by the RUN command. Notice the INPUT command operates in the usual manner by displaying a question mark for each input value required. As soon as you have entered two values, the line number of the INPUT statement appears:

```
1Ø:
```

By pressing the ↑ (Up Arrow) key and holding it, you may review the entire line:

```
1Ø: INPUT A, B
```

* In the trace mode, after the calculated result is displayed at the location specified with the CURSOR command, the next line number is displayed on the following line. (See page 128 for a description of the CURSOR command.)

* In the trace mode, if variables are called or if a calculation is performed manually when the display starting position has been specified with the cursor command, the display starting position will be cleared.

CHAPTER 11
MAINTENANCE OF THE PC-2500

To insure trouble-free operation of your SHARP PC-2500 we recommend the following:

* Always handle the portable computer carefully as the liquid crystal display is made of glass.

* Keep the computer in an area free from extreme temperature changes, moisture, or dust. During warm weather, vehicles left in direct sunlight are subject to high temperature build up. Prolonged exposure to high temperature may cause damage to your computer.

* Use only a soft, dry cloth to clean the computer. Do not use solvents, water, or wet cloths.

* If service is required, the computer should only be returned to a SHARP SERVICE DEALER.

* If the computer is subjected to strong static electricity or external noise it may "hang up" (all keys become inoperative). If this occurs, press the ALL RESET button while holding down any key. (See Troubleshooting).

* Keep this manual for further reference.

APPENDIX A
ERROR MESSAGES

There are nine different codes built into the PC-2500. The following table will explain these codes.

Error Number **Meaning**

1 Syntax error.

- This means that the PC-2500 can't understand what you have entered. Check for things such as semicolons on the ends of PRINT statements, misspelled words, and incorrect usages.

```
3*/2
```

2 Calculation error

Here you have probably done one of three things:

1. Tried to use too large a number.
   Calculation results are greater than 9.999999999E 99.
2. Tried to divide by zero.

```
5/Ø
```

3. An illogical calculation has been attempted.

```
LN −3Ø  or  ASN 1.5
```

3 Illegal Function (DIMension error/Argument error)

- Array variable already exists.
  Array specified without first dimensioning it.
  Array subscript exceeds size of array specified in DIM statement.

```
DIM B(256)
```

- Illegal function argument. This means that you have tried to make the computer do something that it just can't handle.

```
WAIT 66ØØØ
```

4 Too Large A Line Number

Here you have probably done one of two things:

1. Tried to use a non-existent line number by the GOTO, GOSUB, RUN, LIST or THEN etc.
2. Tried to use too large a line number. The maximum line number is 65279.

5 Next Without A For ...

Subroutine nesting exceeds 1Ø levels.

FOR loop nesting exceeds 5 levels.
RETURN verb without a GOSUB, NEXT verb without a FOR, or READ verb without a DATA.

Buffer space exceeded.

6 Memory Overflow.
Generally this error happens when you've tried to DIMension an array that is too big for memory. This can also happen when a program becomes too large.
- ● The reserve content exceeds 144 bytes.

7 PRINT USING error.
This means that you have put an illegal format specifier into a USING statement.

8 I/O device error.
This error can happen when you have the optional printer and/or cassette recorder connected to the PC-2500. This error can also happen when you use the serial input/output. It means that there is a problem with communication between the I/O device and the PC-2500.

Low Battery
This means that the printer cannot be operated because the voltage of the built-in rechargeable batteries is low.

9 Other errors.
This code will be displayed whenever the computer has a problem that isn't covered by one of the other eight error codes. One of the most common causes for this error is trying to access data in a variable is one fashion (e.g. A$) while the data was originally stored in the variable in another fashion (e.g. A).

Regarding Input Errors
When executing a program, an error may occur due to input errors of the program In this case note the following.
Example: When KPRINT is entered instead of LPRINT

```
                 1Ø: K PRINT A$
          →      1Ø █ PRINT A$
      L ENTER    1Ø L PRINT A$
                    ↑
                    └──Space
```

When corrected in this manner, the computer does not recognize it as a command. In this sample, erase KPRINT and re-enter LPRINT.

```
                                  10: K PRINT A$
                             →    10 ■ PRINT A$
SHIFT  +  DEL  SHIFT  +  DEL      10 ■$
                  INS ... INS     10 ■_ _ _ _ _ A$
                LPRINT ENTER      10: LPRINT A$
                                      ↑
```

Space is not needed when recognized as a command.

The command can be checked to see if it is entered correctly by using the cursor key.

(Correct input)

```
    10: RADIAN
→   10 ■ADIAN
→   10 RADIAN_
```

(Wrong input)

```
    10: RADAN
→   10 ■ADAN
→   10 R■DAN
→   10 RA■AN
```

APPENDIX B CHARACTER CODE CHART

The following chart shows the conversion values for use with CHR$ and ASC. The column shows the first hex character or the first four binary bits, the row shows the second hex character or the second binary bits. The upper left corner of each box contains the decimal number for the character. The lower right shows the character. If no character is shown then it is an illegal character on the PC-2500.

For example, the character "A" is a decimal 65 or a hex 41 or binary Ø1ØØØØØ1. The character '√‾' is a decimal 252 or a hex FC or a binary 111111ØØ.

The character codes are represented as follows.
Example

- Code for "*"
  Hexadecimal &2A
  Decimal 42 (2 * 16 + 1Ø)
- Code for "√‾"
  Hexadecimal &FC
  Decimal 252 (15 * 16 + 12)

Notes:

- Displaying a character using the CHR$ command:
  The character for code Ø (&ØØ) in the table is null. Nothing is displayed. Squares in the table where no characters are listed are spaces.

* Printing characters on the built-in printer using the CHR$ command:

- The following codes in codes 9 (&ØØ) to 31 (&1F) are control codes. The other codes are null codes.

  8 (&Ø8) . . . BS
  1Ø (&ØA) . . . LF
  11 (&ØB) . . . LU
  13 (&ØD) . . . CR
  27 (&1B) . . . ECR

  Squares in the table where no characters are listed are printed as spaces.

- Codes 249 (&F9) and 25Ø (&FA) are spaces.

First 4 Bits

Second 4 Bits

Hex Binary	0 0000	1 0001	2 0010	3 0011	4 0100	5 0101	6 0110	7 0111	8 1000	E 1110	F 1111
0 0000	0 NUL	16	32 SPACE	48 0	64 @	80 P	96 `	112 p	128	224	240
1 0001	1	17	33 !	49 1	65 A	81 Q	97 a	113 q	129	225	241
2 0010	2	18	34 "	50 2	66 B	82 R	98 b	114 r	130	226	242
3 0011	3	19	35 #	51 3	67 C	83 S	99 c	115 s	131	227	243
4 0100	4	20	36 $	52 4	68 D	84 T	100 d	116 t	132	228	244
5 0101	5	21	37 %	53 5	69 E	85 U	101 e	117 u	133	229	245 ♠
6 0110	6	22	38 &	54 6	70 F	86 V	102 f	118 v	134	230	246 ♥
7 0111	7	23	39 '	55 7	71 G	87 W	103 g	119 w	135	231	247 ♦
8 1000	8	24	40 (	56 8	72 H	88 X	104 h	120 x	136	232	248 ♣
9 1001	9	25	41)	57 9	73 I	89 Y	105 i	121 y	137	233	249 █
A 1010	10	26	42 *	58 :	74 J	90 Z	106 j	122 z	138	234	250 Σ
B 1011	11	27	43 +	59 ;	75 K	91 [	107 k	123 {	139	235	251 ×
C 1100	12	28	44 ,	60 <	76 L	92 \	108 l	124 ¦	140	236	252 √
D 1101	13	29	45 _	61 =	77 M	93]	109 m	125 }	141	237	253
E 1110	14	30	46 .	62 >	78 N	94 ^	110 n	126 ~	142	238	254
F 1111	15	31	47 /	63 ?	79 O	95 —	111 o	127	143	239	255

APPENDIX C
FORMATTING OUTPUT

It is sometimes important or useful to control the format as well as the content of output. The PC-2500 controls display formats with the USING verb. This verb allows you to specify:

- The number of digits
- The location of the decimal point
- Scientific notation format
- The number of string characters

These different formats are specified with an "output mask." This mask may be a string constant or a string variable:

```
1Ø: USING "####"
2Ø: M$="&&&&&&"
3Ø: USING M$
```

When the USING verb is used with no mask, all special formatting is cancelled.

```
4Ø: USING
```

A USING verb may also be used within a PRINT statement:

```
5Ø: PRINT USING M$; N
```

Wherever a USING verb is used, it will control the format of all output until a new USING verb is encountered.

Numeric Masks

A numeric USING mask may only be used to display numeric values, i.e., numeric constants or numeric variables. If a string constant or variable is displayed while a numeric USING mask is in effect, the mask will be ignored. A value which is to be displayed must always fit within the space provided by the mask. The mask must reserve space for the sign character, even when the number will always be positive. Thus a mask which shows four display positions may only be used to display numbers with three digits.

Specifying Number of Digits

The desired number of digits is specified using the '#' character. Each '#' in the mask reserves space for one digit. The display or print always contains as many characters as are designated in the mask. The number appears to the far right of this field; the remaining positions to the left are filled with spaces. Positive numbers therefore always have at least one space at the left of the field. Since the PC-2500 maintains a maximum of 1Ø significant digits, no more than 11 '#' characters should be used in a numeric mask. When the total number of columns of the integer part specified exceed 11, this integer part is regarded as 11 digits in the PC-2500.

Note: In all examples in this appendix the beginning and end of the displayed field will be marked with a 'I' character to show the size of the field.

Statement	Display
1Ø: USING "####"	(Set the PC-2500 to the RUN mode, type RUN, and press ENTER .)
2Ø: PRINT 25	I 25 I
3Ø: PRINT −35Ø	I −35Ø I
4Ø: PRINT 1ØØØ	ERROR 7 IN 4Ø

Notice that the last statement produced an error because 5 positions (4 digits and a sign space) were required, but only 4 were provided in the mask.

Specifying a Decimal Point

A decimal point character, '.', may be included in a numeric mask to indicate the desired location of the decimal point. If the mask provides more significant decimal digits than are required for the value to be displayed, the remaining positions to the right will be filled with zeros. If there are more significant decimal digits in the value than in the mask, the extra digits will be truncated (not rounded):

Statement	Display
1Ø: USING "####.##"	
2Ø: PRINT 25	I 25.ØØ I
3Ø: PRINT −35Ø.5	I −35Ø.5Ø I
4Ø: PRINT 2.547	I 2.54 I

Specifying Scientific Notation

A "^" character may be included in the mask to indicate that the number is to be displayed in scientific notation. The '#' and '.' characters are used in the mask to specify the format of the "characteristic" portion of the number, i.e., the part which is displayed to the left of the E. Two '#' characters should always be used to the left of the decimal point to provide for the sign character and one integer digit. The decimal point may be included, but is not required. Up to 9 '#' characters may appear to the right of the decimal point. Following the characteristic portion, the exponentiation character, E, will be displayed followed by one position for the sign and two positions for the exponent. Thus, the smallest scientific notation field would be provided by a mask of "##^" which would print numbers of the form '2E 99'. The largest scientific notation field would be "##.#########^" which would print numbers such as '–1.23456789ØE–12':

```
Statement                               Display

1Ø: USING "###.##^"

2Ø: PRINT 2                             |  2.ØØE ØØ|

3Ø: PRINT –365.278                      |–3.65E Ø2|
```

Specifying Alphanumeric Masks

String constants and variables are displayed using the '&' character. Each '&' indicates one character in the field to be displayed. The string will be positioned at the left end of this field. If the string is shorter than the field, the remaining spaces to the right will be filled with spaces. If the string is longer than the field, the string will be truncated to the length of the field:

```
Statement                               Display

1Ø: USING "&&&&&&"

2Ø: PRINT "ABC"                         |ABC   |

3Ø: PRINT "ABCDEFGHI"                   |ABCDEF|
```

Mixed Masks

In most applications a USING mask will contain either all numeric or all string formatting characters. Both may be included in one USING mask, however, for certain purposes. In such cases, each switch from numeric to string formatting characters or vice versa masks the boundary for a different value. Thus, a mask of "#####&&&&" is a specification for displaying two separate values—a numeric value which is allocated 5 positions and a string value which is allocated 4 positions:

```
 Statement                                 Display

1Ø: PRINT USING "###.##&&";25;"CR"        |25.ØØCR|

2Ø: PRINT –5.789;"DB"                     |–5.78DB|
```

Remember: Once specified, a USING format is used for all output which follows until cancelled or changed by another USING verb.

APPENDIX D EXPRESSION EVALUATION AND OPERATOR PRIORITY

When the SHARP PC-2500 is given a complex expression, it evaluates the parts of the expression in a sequence which is determined by the priority of the individual parts of the expression. If you enter the expression:

1ØØ/5+45

as either a calculation or as a part of a program, the PC-2500 does not know whether you mean:

$\frac{100}{5+45} = 2$ or $\frac{100}{5} + 45 = 65$

Since the PC-2500 must have some way to decide between these options, it uses its rules of operator priority. Because division has a higher "priority" than addition (see below), it will choose to do the division first and then the addition, i.e., it will choose the second option and return a value of 65 for the expression.

Operator Priority

Operators on BASIC of the SHARP PC-2500 are evaluated with the following priorities from highest to lowest:

Level	Operations
1.	Parentheses
2.	Variables and Pseudovariables
3.	Functions
4.	Exponentiation (^)
5.	Unary minus, negative sign (–)
6.	Multiplication and division (*, /)
7.	Addition and subtraction (+, –)
8.	Relational operators (<, <=, =, <>, >=, >)
9.	Logical operators (AND, OR, NOT)

When there are two or more operators at the same priority level the expression will be evaluated from left to right. (The exponentiation will be evaluated from right to left). Note that with A+B–C, for example, the answer is the same whether the addition or the subtraction is done first.

When an expression contains multiple nested parentheses, the innermost set is evaluated first and evaluation then proceeds outward.

For level 3 and 4, the last entry has a higher priority.

For example: −2 ^ 4 → −(2^4)
3 ^ −2 → 3^{-2}

Sample Evaluation

Starting with the expression:

((3+5−2)*6+2)/1Ø^ LOG 1ØØ

The PC-2500 would first evaluate the innermost set of parentheses. Since '+' and '−' are at the same level it would move from left to right and would do the addition first:

((8−2)*6+2)/1Ø^ LOG 1ØØ

Then it would do subtraction:

((6)*6+2)/1Ø^ LOG 1ØØ

or:

(6*6+2)/1Ø^ LOG 1ØØ

In the next set of parentheses it would do the multiplication first:

(36+2)/1Ø^ LOG 1ØØ

And then the addition:

(38)/1Ø^ LOG 1ØØ

or:

38/1Ø^ LOG 1ØØ

Now that the parentheses are cleared, the LOG function has the highest priority so it is done next:

38/1Ø^2

The exponentiation is done next:

38/1ØØ

And last of all the division is performed:

Ø.38

This is the value of the expression.

APPENDIX E
KEY FUNCTIONS IN BASIC

SHIFT : This key switches the function for keys having 2 functions.

- If this key is held down while an alphabet key is pressed, an upper case letter is entered. (Lower case letters are entered if the CAPS key lamp is on.)
- If this key is held down while a key having a number and a symbol written on it is pressed, the symbol written at the top of the key is entered.

CAPS : Sets and clears the upper case letter mode. (This key also turns on and off the CAPS key lamp and displays and clears the CAPS symbol.)
The entry of upper case and lower case letters is reversed when the CAPS key is pressed.

A to Z : Alphabet Keys
Enters the alphabet. Press the SHIFT key or CAPS key to switch from lower case to upper case letters.

□ : Use to provide space when entering program or characters.

!1 @2 ~)0 : Press these keys to enter numbers. Hold the SHIFT key down and press these keys to enter the symbols written at the top of the keys.

@ : • Used for reserve contents when the reserve key is used as a program key.
Example: GOTO 100@

: • Use with USING statement, to provide the instruction to define the display format of numerical data.

$: • Use when assigning character variables.

^ : • Use for power calculation instructions.
• Use to specify the exponent display system for numerical data in USING statement instructions.

& : • Use with USING statement, to provide the instruction to define the display format of character string.
• Use to designate hexadecimal number.

* : • Use for the multiplication command.

( ) : • Use for parentheses in equations.

! % : Used as a character string within " ".

[=] [±] [″/] [;] [?/] [:] [<.] [>.] :

Press these keys to enter the symbols written at the bottom of the keys. Hold the [SHIFT] key down and press these keys to enter the symbols written at the top of the keys.

— : Use for the subtraction command or the minus sign.

= : • In assignment statements, use to assign the contents (number or character) on the right for the variable specified on the left.
• Use when entering logical operators in IF sentence.

+ : Use for the addition command.

" : • Use to designate and cancel characters.
• Use to specify labels.

? : Use to enter CLOAD?

/ : Use for the division command.

; : Use to separate equations, variables, comments, a command and a variable, etc.

: : Use to divide two or more statements in one line.

, : Use to provide pause between two equations, and between variables or comments.

<> : Use when entering logical operators in IF sentence.

. : • Decimal point.
• Use to enter an abbreviation of a command/verb/function.
• Use to designate the decimal portion in USING format designation.

_ ' [ ] : Used as a character string within " ".

[0] to [9] : Used to enter numbers or numeric values.

. : • Use to enter the decimal point.
• Use to enter an abbreviation of a command/verb/function.
• Use to designate the decimal portion in USING format designation.

— : Use to enter the subtraction command or the minus sign.

+ : Use to enter the addition command or the plus sign.

* : Use to enter the multiplication command.

/ : Use to enter the division command.

[INS] : Inserts one space (⌶ appears) of 1-step capacity between the address (N) indicated by the cursor and the preceding address (N—1).

DEL BS :

- Use to delete the character on the left of the cursor. The character at and to the right of the cursor are moved to the left by one character.
- If the cursor is at the beginning of a line, the character at the cursor is deleted.
- If the SHIFT key is held down while this key is pressed, the character or command at the cursor is deleted.

ENTER :

- Specifies the end of a program line.
- Writes the program or reserve contents.
- Executes manual calculations or manual operations for BASIC commands.
- Executes the program (e.g. restarts the program stopped temporarily by the INPUT command or PRINT command).

ON/BRK :

(ON)

Use to turn the PC-2500 power on when the auto power off function is in effect.

(BREAK)

- Depressing this key during program execution functions as a BREAK (ON/BRK) key and causes the program to interrupt execution.
- When pushed during manual execution, input/output command such as BEEP, CLOAD, etc., execution of the command is interrupted.

SHIFT + ON/BRK : Calls the menu screen for the Business Software and BASIC selection.

CLS

- Use to clear the contents of the entry and the display.
- Use to reset the error.

SHIFT + CLS :

- Not only clears the display contents, but resets the computer to its initial state.

  — Initial state —

  - Resets the WAIT timer.
  - Resets the display format. (USING format)
  - Resets the TRON state (TROFF).
  - Resets the PRINT=LPRINT.
  - Resets error.

MODE :

- Use the change the operational mode selection from RUN to PROgram or from PROgram to RUN.

SHIFT + MODE :

- Use to set the reserve mode.

→ :

- Shifts the cursor to the right (press once to advance one position, hold down for automatic advance)
- Executes playback instructions.
- Call the cursor if not displayed while the contents are displayed.

- Clears an error condition in manual operation.
- Moves the index in the Business Software.

[SHIFT]+[→] :
- Moves the cursor to the last column in the program line or entered contents when the cursor is displayed.
- Same as the [→] key when the cursor is not displayed.

[←] :
- Shifts the cursor to the left (press once to advance one position, hold down for automatic advance)
- Otherwise the same an the [→] key.

[SHIFT]+[←] :
- Moves the cursor to the first column in the program line or entered contents when the cursor is displayed.
- Same as the [←] key when the cursor is not displayed.

[DEF] :
- Executes a defined program or calls the reserve contents.
- Enters the various modes in the Business Software.

[PEN] :
- Rotates the pen holder so that the pens can be replaced.

[SHIFT]+[PEN] :
- Sets the pen replace mode.
- Clears the pen replace mode.

The following describes the effective keys while in the Business Software.

[DEF] [PEN] (PROGRAM) : Enters the table creation mode. A table can be created by entering an "expression."

[DEF] [←] (FORMAT) : Enters the display format set mode and enables you to
- Set the total column and average column
- Set the notation for numeric values
- Sort data
- Check the "expression"

[DEF] [↑] : Enters the data write mode.
Appends, modifies, and deletes data in the table.

[DEF] [↓] (SELECT) : Enters the table selection mode.
Selects a stored table. The table selected here can be printed or used to create a graph.

[DEF] [ON/BRK] (SYSTEM) : Special function execution mode enables you to
- Check the remaining memory
- Delete a table
- Delete all tables
- Set the functions for the RAM card

DEF → : Enters the table transfer mode and enables you to

- Store table data to tape, and load or verify table data from tape
- Perform table data input/output operations using the serial I/O interface

DEF CLS : Enters the print mode and enables you to print

- Tables
- All titles
- An expression
- All expressions

DEF MODE : Enters the graph creation mode.

MODE : Used to search data when table data is displayed.

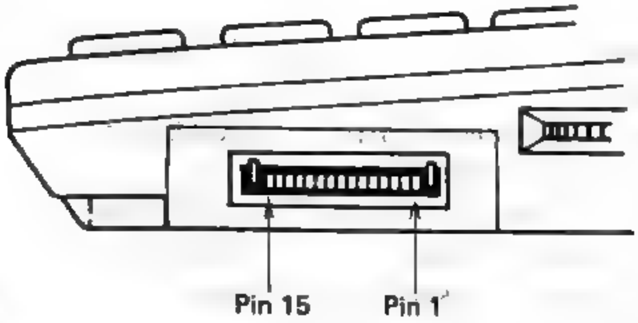
The ↑ and ↓ keys have the following functions, depending on designated modes, as well as the state of the computer.

Mode	State	↓	↑
RUN	Program being executed		
	Program is temporarily interrupted	To execute the next line	To display program line being executed or already executed, hold this key down.
	PRINT, LINE, PRESET, PSET, GPRINT statement being executed		
	PRINT statement just now executed		
	Under break		
	INPUT statement being executed	non function	
	Error condition during executing program		To display error-producing line, hold this key down.
	TRON condition	To execute debugging operation	To display program line being executed or already executed, hold this key down.
PRO	(When the mode is changed from RUN to PRO and program line is not being displayed)		
	Program is temporarily interrupted	To display the line interrupted	Same as left.
	Error condition	To display the line with error	Same as left
	Other condition	To display the first line	To display the last line
	(When the program line is being displayed)		
		To display the next program line	To display the preceeding program line
RESERVE			

- On the display, the ENTER key is the same as a space.
- If no key is entered in the key input request mode for approximately 11 minutes, the power is automatically turned off (auto-power off function).

APPENDIX F SIGNALS USED IN THE SERIAL I/O TERMINAL

The PC-2500 is equipped with a 15-pin connector for the serial I/O terminal. The pins used and their signals are described below.



PC-2500
Right side view

Pin Connections Used

Pin	Name	Symbol	I/O	Function
1	Frame Ground	FG		Ground for maintenance
2	Transmit Data	SD	o	DC output signal
3	Receive Data	RD	i	DC input signal
4	Request Data	RS	o	ON: sends carrier
5	Clear To Send	CS	i	ON: transmission enabled
7	Signal Ground	SG	—	Reference 0 voltage for all signals
8	Data Carrier Detect	CD	i	ON: carrier signal received
10		VC		Power supply
11	Receive Ready	RR	o	ON: receive enabled
13		VC		Power supply
14	Data Terminal Ready	ER	o	ON: local terminal ready

Note: The voltage level of VC is the high level. The voltage level of SG is the low level.

Note: If a voltage which exceeds the allowable range (voltage level between SG and VC) of the PC-2500 is applied, the internal components may become damaged since the PC-2500 uses CMOS components.

APPENDIX G
SPECIFICATIONS

Model:	PC-2500 Portable Computer	
Processor:	8 bit CMOS CPU	
Programming Language:	BASIC	
System ROM:		72 K Bytes
Memory Capacity:	RAM:	
	System	1.7 K Bytes
	User	
	Fixed Memory Area (A ~ Z, A$ ~ Z$)	2Ø8 Bytes
	Program/Data Area	31Ø2 Bytes
	Reserve Area	79 Bytes
Stack:	Sub-routine: 1Ø stacks	Function: 16 stacks
	FOR-NEXT: 5 stacks	Data: 8 stacks
Operators:	Addition, subtraction, multiplication, division, trigonometric and inverse trigonometric functions, logarithmic and exponential functions, angle conversion, square and square root, sign, absolute, integer, relational operators, logical operators.	
Numeric Precision:	1Ø digits (mantissa) + 2 digits (exponent).	
Editing Features:	Cursor left and right, line up and down, Back space, character insert, character delete.	
Memory Protection:	CMOS Battery backup.	
Software:	SHARP Business Software • Tabular calculation • Graph function: Bar, broken→ line, band circle graph • Telephone book	

PRINTER

Type:	X/Y axis plotter	
Color plotting:	4 colors (black, blue, green, red standard pen set)	
Number of character sizes:	15 types from 0.8 x 1.2 mm (0.032 x 0.048 in.) to 12 x 18 mm (0.48 x 0.72 in.) (Standard characters)	
Minimum pen movement:	0.2 mm (0.008 in.)	
Plotting while drawing characters:	Max. 7 characters/sec. (standard character size "b")	
Paper:	Roll paper:	
	Outer diameter	Less than 25 mm (1 in.)

	Inner diameter	8 mm (0.3 in.)
	Width	114 mm (4.5 in.)
	(EA-515P, option)	

Memory Protection: CMOS Battery backup
(Program, data and reserved contents are protected when the power is OFF.)

Pen movement speed:	X and Y directions	73 mm/sec.
	45° angle	103 mm/sec.

Serial Input/Output Features:	Standards:	Start-stop transmission (asynchronous) system.
	Baud Rates:	3ØØ, 6ØØ, 12ØØ Baud
	Data Bits:	7 or 8 Bits
	Parity Bits:	Even, odd, or no-parity
	Stop Bit:	1 or 2 Bits
	Connectors Used:	15-pin connector (for external equipment)
	Output Signal Level:	C-MOS level (4—6 Volts)
	Interfacing Signals:	Inputs: RD, CS, CD Outputs: SD, RS, RR, ER Others: SG, FG, VC

Display: 4-line 24-digit liquid crystal display with 5 x 7 dot characters or 15Ø x 32 dot graphics.

Power Supply: Built-in rechargeable Ni-Cd battery

Power Consumption: 6V ⎓ (DC): 6W

- Approx. 60 hours (operation at 2Ø°)
  This assumes that out of each hour, the program execution and calculation time is 1Ø minutes and the display time is 5Ø minutes. This does not include printer operation.
- Approx. 45Ø printing lines (continuous printing at 2Ø°C where each line contains 2Ø digits of 5s in standard character size "b")
- Approx. 11 printed graphs (when the graph shown on page 304 is continuously printed)

Operating Temperature: 5°C to 4Ø°C (41°F to 1Ø4°F)

Dimensions: 297(W) x 21Ø(D) x 45.5(H) mm
11-11/16"(W) x 8-1/2"(D) 1-25/32"(H)

Weight: Approximately 1.3 kg (2.78 lbs.)

Accessories: Tape recorder connecting cord, AC adapter (EA-15Ø), pens (black, blue, green, red, 1 each), roll paper (1 roll), and operation manual.

Options: Plug-in Card type 8 K Bytes RAM (CE-2Ø1M),
16 K Bytes RAM (CE-2Ø2M)
Cassette Tape Recorder (CE-152) etc.

APPENDIX H
USING PROGRAM DEVELOPED FOR THE PC-1210 SERIES, PC-1245 SERIES, PC-1260 SERIES, PC-1350 OR PC-1401

Note: PC-1210 Series: PC-1210, PC-1211
PC-1245 Series: PC-1245, PC-1250, PC-1251
PC-1260 Series: PC-1260, PC-1261

The display units in the PC-1245 Series, PC-1210 Series, PC-1260 Series and PC-1401 Pocket Computers consist of either 1 or 2 lines. As a result, to change the contents shown on the display unit, the previous contents must first be cleared. In the PC-2500, however, the display contents (4 lines) scroll up with each new line.

Therefore, when executing programs which move characters or symbols, or programs using the CURSOR command in the PC-1260 Series, unexpected movements or displays may result. It is thus necessary to modify the program, such as by clearing the screen each time using the CLS command.

Further, since the functions may differ slightly depending on the series of the pocket computer, the modifications to the program described below are required.

Modifications Required to PC-1245 series (PC-1245, PC-1250, PC-1251) Programs

When using programs developed for the PC-1245 series on the PC-2500, it is necessary to modify the following:

1. Multiplication without using the operator "*".
   On the PC-1245 series, the operator (*) for multiplication may be omitted, such as AB for A*B or CD for C*D. On the PC-2500, the multiplication operator (*) cannot be omitted since the computer treats two consecutive characters, such as AB or CD, as simple variables. Use the specification on the right hand side of the following example:

   (e.g.) A = SIN BC → A = SIN (B*C)

2. Definition of subscripted variables (such as A( )) by using the DIM statement: On the PC-1245 series, if, for example, DIM A(3Ø) is executed, memory locations for A(27) through A(3Ø) are set aside as an extension of a fixed variable definition area. On the PC-2500, however, the execution of DIM A(3Ø) reserves a separate memory area for array variables A(Ø) through A(3Ø) for the array named A. When defining subscripted variables (such as A( )) as an extension of fixed variables, use the specification on the right hand side of the following example:

   DIM A(3Ø) → A(3Ø) = Ø

3. Data I/O statement for tape files:

On the PC-1245 series, the execution of, for instance, the PRINT # C statement saves the contents of the variable C and all the subsequent variables to a tape file. On the PC-2500, however, the execution of the same statement saves the contents of the variable C only. To save the contents of a specific variable and all the subsequent variables, use the specification on the right hand side of the following examples:

```
(e.g.)  PRINT # A  →  PRINT # A*
        PRINT # C  →  INPUT # C*
```

Note: On the PC-2500 you cannot execute programs for the PC-1250/1251 in which the POKE or CALL command is used. Executing such programs on the PC-2500 may cause the abnormal situation (for example, you can get no response from any key).

4. Value of a loop variable after completion of a FOR-NEXT loop:

The value of a loop variable obtained after the execution of a FOR-NEXT loop completed on the PC-2500 is different from that obtained on the PC-1245 series. If the value of a loop variable is used in a conditional expression in a PC-1245 series program, increment it by one when it is used on the PC-2500.

```
(e.g.)   1Ø FOR I=Ø TO 1Ø

         5Ø NEXT I
         6Ø IF I=1Ø THEN 1ØØ
```

Modify the value of I in line 6Ø as follows:

```
         6Ø IF I=11 THEN 1ØØ
```

(On the PC-2500, the value of a loop variable must be incremented by one step value. The number of loop execution cycles remains the same, however.)

5. Exponent symbol "E":

The PC-2500 uses the uppercase letter "E" for its exponent symbol. The following change is required:

```
A = 1.234 E 5  →  A = 1.234E5
B = E 6        →  B = 1E6
```

If a PC-1245 program is read from a tape file into the PC-2500, the change for the exponent symbol described above will automatically be done by the PC-2500.

6. The character codes of the PC-1245 series are partially different from those of the PC-2500.

When the following codes are designated by the CHR$ function, change the codes.

Character Code	PC-1245	PC-2500
39 (&27)	□	'
91 (&5B)	√	[
92 (&5C)	¥	\
93 (&5D)	π	]
96 (&60)	E	`
250 (&EA)	– (Error)	□
251 (&FB)	– (Error)	π
252 (&FC)	– (Error)	√

Note: As shown above, the PC-2500 does not have the character E.

7. Modifications to the defined = key
   The = key is not a defined key in this computer. Therefore, for programs which define the = key, define another key.

   Example:

   ```
   100 "=": → 100 "A":
   ```

Additional Modifications

1. The PC-1245, PC-1250, and PC-1251 use a line number ranging from 1–999, whereas this model, as well as the PC-1260/1261, has an extended line number ranging from 1–65279. Therefore, the line number uses 3 bytes in RAM (PC-1245 series uses 2 bytes). The modification is carried out automatically when the program is loaded through the cassette tape. However, there is a possibility of memory overflow (ERROR 6) when loading or executing a long program. Further, when a single line is close to 80 bytes long, this modification may result in the clearing of the end of the line.

2. In loading a program of the PC-1245 series through the cassette tape, the computer will remain BUSY for one to two seconds after the tape has stopped due to modification of the line number (2 bytes to 3 bytes) as mentioned previously. During this period, symbol "*" will be displayed at the right bottom column of the display as in loading a program.

Note: The PC-1245 series cannot read from a tape which contains programs developed for the PC-2500.

Modifications Required to PC-1401 Series Programs

(1) Modifications to the defined , key
The , key is not a defined key in this computer. Therefore, for programs which define the , key, define another key.

Example:

```
100 ",": → 100 "A":
```

(2) Other notes

The PC-1401 contains more function commands than the PC-2500. Therefore, an error will occur during execution if a program which uses commands not available in the PC-2500, is written or read from a tape. If a program is read from a tape and contains commands not available in the PC-2500, those commands will be substituted with "~" and then displayed.

Modifications Required to PC-1210 Series Programs

To use PC-1210 Series programs on the PC-2500, they must be modified in the same way as PC-1245 Series programs (except items 2 and 6). In addition, the following modifications are necessary.

(1) IF Statement

If, for example,

```
50 IF A> L PRINT "A" (display "A" if A>L)
```

is found in the program for the PC-1210 Series Pocket Computers, it is interpreted as

```
50 IF A>LPRINT "A" (Print out "A" if A>)
```

and results in an error when it is entered though the keyboard.

The error occurs because a command which does not exist in the PC-1210 Series does in fact exist in the PC-2500.

To solve this problem, insert a THEN command into the IF statement as follows.

```
50 IF A> L THEN PRINT "A"
```

(2) Specified Format in USING

The function of the USING command differs between the PC-2500 and the PC-1210 Series as follow.

[Example]

```
1Ø A = –123.456
2Ø PAUSE USING "####.##"; A
3Ø PAUSE A, USING "####"; A
```

Executing this program displays the following.

```
* PC-1210/PC-1211                                  –123.45
                                       –123           –123

* PC-2500                                          –123.45
                                    –123.45           –123
```

For the execution of line 3Ø in the PC-1210 Series, the display on the left side also follows the displayed format on the right side. In the PC-2500, the display follows the previous specified format. This applies not only to the PAUSE command, but also to the PRINT and LPRINT commands.

(3) Omitting ")"
In the PC-1210 Series, the ")" which comes immediately before the ENTER or: (colon) can be omitted. It cannot be omitted in the PC-2500. Therefore, be sure to add the ")" to the program if omitted.

(4) Print Command
The PC-2500 has a PRINT command for displays and a LPRINT command for printing. However, all PRINT commands can be used for printing if PRINT = LPRINT is specified (See page 155.)
The PC-1210 Series does not have the LPRINT command. To print using a PC-1210 Series program, add PRINT = LPRINT to the program or, execute manually.

(5) Variables
When the RUN command has been executed in the PC-1210 Series, all variables are retained. In the PC-2500, however, all variables from A(27) and on are cleared. (See page 47).
Therefore, if there is a need to retain variables at the start of program execution, start the program execution using the GOTO command or function defined keys.

Modifications Required to PC-1260 Series Programs

(1) Modifying the Character Code
In the PC-1260 Series, the character for character code 96 (&6Ø) is a space, whereas in the PC-2500 it is a ‘ (single quote).
Therefore, if a space has been specified using code 96 in the CHR$ command, change it to code 32 ($2Ø)

(2) Modifications to the defined ≡ key
The ≡ key is not a defined key in this computer. Therefore, for programs which defined the ≡ key, define another key.

Example:

```
    1ØØ "=": → 1ØØ "A":
```

Modifications Required When Using PC-1350 Program

(1) **Printing Columns (program using the CE-126P)**
The optional CE-126P printer prints 24 columns per line. The built-in printer of the PC-2500 usually prints 39 columns per line.
Therefore, this difference will cause the printed positions to shift even though the same LPRINT command is used.
To avoid this
① Change the LPRINT command
② Insert the

```
            CONSOLE 24: LPRINT
```

command at the beginning of the program to set the printing line to 24 columns.

(2) Redefining the [=] Key

The same modification shown on page 240 is required.

(3) Changing the Character Code

The character for character code 96 (&6Ø) in the PC-1350 is ' while it is ' in the PC-2500.

If character ' is not desired when code 96 (&6Ø) is specified in the CHR$ command, specify another character.

(4) Using the CONSOLE Command

Change the CONSOLE command to the form

CONSOLE expression: LPRINT

for programs to be executed on the PC-2500.

Example:

```
   :                         :
 1ØØ CONSOLE 36            1ØØ CONSOLE 36: LPRINT
 11Ø LPRINT A, B, C        11Ø LPRINT A, B, C
   :                         :
```

(5) Entering π and √‾

The characters π and √‾ cannot be entered through the keyboard on the PC-2500.

Use PI for π and SQR for √‾ When entering programs through the keyboard.

EXAMPLES OF PROGRAMMING

Having studied the explanations of each function in the preceding chapters up to this page, we are sure that you have acquired a broad knowledge of a great deal of program commands. However, in order for you to freely create the application programs at your disposal with BASIC language, it is absolutely necessary for you to try to run your own programs in addition to those in this operation manual. Just as you improve your driving by holding the steering wheel or improve your tennis by swinging the racket, you attain proficiency in programming by practicing as much as possible regardless of your own ability. It will also be a great help if you refer to programs that have been done by other people. We will describe in the following chapters some of the programs with each command in "BASIC" for your reference.

(Sharp Corporation and/or its subsidiaries assume no responsibilities or obligations for any losses or damages that should accrue from using these examples of software programs described in the following chapters.)

CONTENTS

(Program title) (Page)

- The display area of the PC-2500 indicates a display of 4 lines, but for the key operation sequence of some programs, only the portions of the display that actually appear are mentioned.
- In the last column of each program, the number of bytes used for each program is described.

Program Title: SKI JUMP

Let's attempt the longest jump!

Today there is fine weather, ideal for ski-jumping. You are enthusiastic about the set-up of the longest jump distance. Jump with perfect timing after level skiing, keeping your balance despite wind from the right and left sides.

Now, let us see how many meters you can make!

■ HOW TO PLAY

1. R U N ENTER displays the title "The Longest Distance". Press the space key and then the game starts.
2. After the jumping stand is displayed on the screen, the man starts skiing. During the course of his level skiing, press the key 5 to let him jump.
   If the jumper fails to jump due to a failure of timing, "0m" will be displayed on the screen and then the game returns to the initial display of the title.
3. Although the skier has jumped, keep the jumper's balance despite wind from the left and right sides by using the following keys.

   Wind from left side (→): Press key 4
   Wind from right side (←): Press key 6

   (The jumper's balance is indicated by the display of a vertical bar "I".)

   - When the jumper lands properly on the ground, the distance covered is displayed, but when he falls halfway due to losing his balance, "Fallen" is displayed on the screen.
     In such a case, no display of the flying distance will appear on the screen.

     When the above procedure is over, the initial display of the game will reappear. Now, you should wait to input the game if you wish to try again.

■ REFERENCE

(The method of calculating the distance covered)

1. Timing of the jump
   J (the descending speed of the jumper depending on the timing) is calculated through the X coordinate (125 – 110) when pressing key 5 .

$$J = (X - 110) \times 0.02$$

Move the jumper along each dot up to X-coordinates 105 – 80. As in every movement, the jumper decreases in accordance with the formula of "Y = Y + J" (Y is for the altitude).

In short, the timing of pressing key $\boxed{5}$ affects the altitude at the point of the X-coordinates 80.

2. Balance

The altitude decreases at a regular speed. The X-coordinates are calculated by the following formula.

$$X = X - (5 - \mathrm{ABS}(B)) \times 0.2$$

B shows the balance and covers the range of the figures from –5 to +5. (When B is above or below the figure, the jumper falls down.)

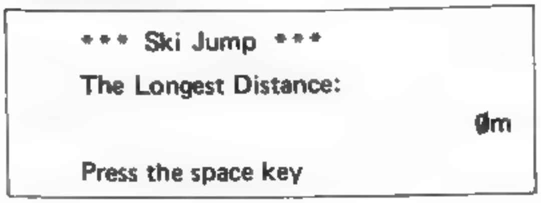
Calculation and display are repeated with the above sequence and then S (flying distance) is calculated from the X-coordinates when the landing judgement formula "$Y > 31 - \mathrm{INT}(X \times 0.05)$" is established.

$$S = (100 - X) \times 1.5$$

Therefore, keeping the proper balance is the knack of increasing the distance covered in flight.

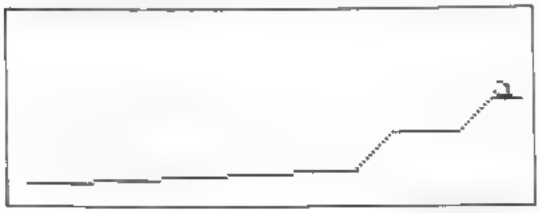
■ KEY OPERATION SEQUENCE

1. [R] [U] [N] [ENTER] [Program starts at]

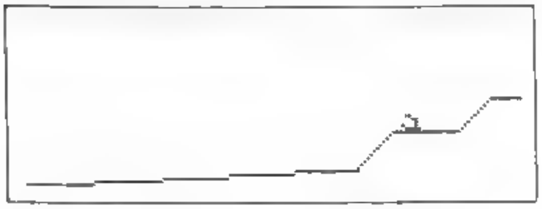


```
     *** Ski Jump ***
     The Longest Distance:
                                   Øm
     Press the space key
```

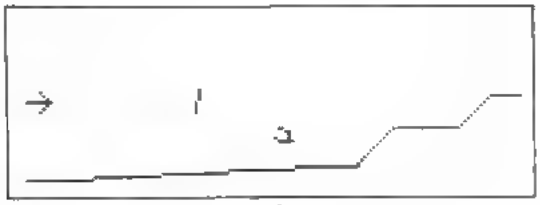
2. [ ] [The game starts]



3. [5]

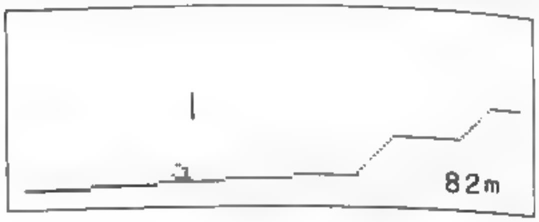


4. [4]



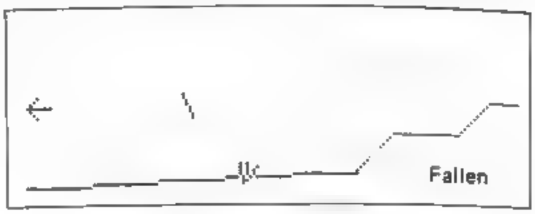
Continue the game by operating 4 or 6 key.

《display example of the well landing》



82m

《display example in the case of falling halfway》



Fallen

■ PROGRAM LIST

```
10:WAIT 0:H=0
20:F1$="40808890F08000":F2$="40808890F
   0000000":F3$="1EE05E201824"
30:DIM S$(5)*16:S$(1)="030C30C00000000
   0":S$(2)="00000FF000000000"
40:S$(3)="000000FF000000":S$(4)="00000
   0F00F000000":S$(5)="00000000C0300C0
   3"
50.CLS : WAIT 0: CURSOR 1,0: PRINT "**
   * Ski Jump ***": BEEP 1
60:CURSOR 0,1: PRINT "The Longest Dist
   ance: ";
70:CURSOR 17,2: PRINT USING "####";H;"
   m";
80:CURSOR 0,3: PRINT "Press the space
   key";
90:I$= INKEY$ : IF I$<>" " THEN 90
100:CLS : LINE (145,5)-(140,5): LINE -(
   130,15): LINE -(110,15): LINE -(99,
   20): GOTO 120
110:FOR I=2 TO 5: LINE (119-I*20,20+I)-
   (100-I*20,20+I): NEXT I: RETURN
120:FOR I=1 TO 5: LINE (119-I*20,20+I)-
   (100-I*20,20+I): NEXT I
130:GCURSOR (141,4): GPRINT F1$:
   GCURSOR (141,4): BEEP 3: GPRINT "00
   0000000000"
140:FOR I=0 TO 10: GCURSOR (134-I,4+I):
   GPRINT F1$: NEXT I
150:X=125
160:I$= INKEY$ : IF I$="5" THEN 220
170:X=X-2: GCURSOR (X,14): GPRINT F2$:
   IF X>110 THEN 160
180:GCURSOR (X,14): GPRINT "0000000000
   0"
190:FOR I=0 TO 11: GCURSOR (103-I,15+I)
   : GPRINT F1$: NEXT I
200:CURSOR 19,3: WAIT 160: PRINT "0 m";
210:GOTO 50
220:J=(X-110)*.02: FOR I=X TO 105 STEP
   -2: GCURSOR (I,14): GPRINT F2$:
   NEXT I
230:Y=14: FOR I=105 TO 80 STEP -1:Y=Y+J
   : GCURSOR (I,Y): GPRINT F1$: NEXT I
   :B=0,X=80
240:R= RND 3: ON R GOTO 250,260,270
250:GCURSOR (0,10): GPRINT "1010101092
   543810":B=B+1: GOTO 280
260:GCURSOR (0,10): GPRINT "0020000000
   000000": GOTO 280
270:GCURSOR (0,10): GPRINT "1038549210
   101010":B=B-1
280:I$= INKEY$
290:IF I$="4" LET B=B-1
300:IF I$="6" LET B=B+1
310:IF B<-5 OR B>5 THEN 400
320:GCURSOR (50,10): GPRINT S$(B*.5+3)
330:X=X-(5- ABS (B))*.2:Y=Y+.26: IF Y>3
   1- INT (X*.05) THEN 370
340:GCURSOR (X,Y): GPRINT F2$
350:IF Y>27 LET C= POINT (X,Y+1): IF C
   GOSUB 110
360:GOTO 240
370:S=(100-X)*1.5: CURSOR 18,3: PRINT
   USING "####";S;: WAIT 320: PRINT "m
   "
380:IF S>H LET H=S
390:GOTO 50
400:FOR I=Y TO 31: GCURSOR (X,I):
   GPRINT F3$: NEXT I
410:CURSOR 17,3: WAIT 160: PRINT "Falle
   n";
420.GOTO 50
```

1279 bytes

■ MEMORY CONTENTS

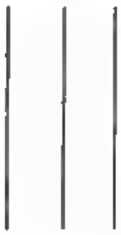
B	balance
C	√
H	The longest flying distance
I, I$	loop counter, √
J	descending speed
R	direction of wind
S	flying distance
X	level distance
Y	altitude
F1$	for display (figure for man)
F2$	for display (figure for man)
F3$	for display (figure for man)
S$(5)	for display (balance)

Program Title: **AMIDAKUJI**

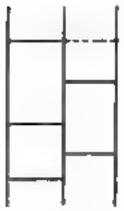
Challenge to a Lottery!

This program has been written to execute a lottery with the aid of a portable-computer. However, let us give you some information about the lottery which we have adopted in this program before proceeding to the subsequent operation. The lottery in question for this program is not the one commonly used, but another type of lottery commonly practiced in Japan called AMIDAKUJI (hereafter refer to this as Amidakuji for this program). Simply, the rules of Amidakuji are just the same as when you guess heads or tails for a coin thrown in the air. For example, with the application of this Amidakuji as another form of lottery, the Amidakuji is often used to give the prize to a person who draws a specially marked line out of many people competing against each other. In ordinary cases, many lines are drawn on the paper but this program will display on the screen the drawing of lines in Amidakuji by means of inputting the numbers of the participants and numbers of the winning mark " * ".

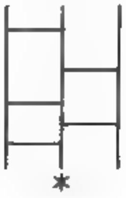
The Rules of Amidakuji



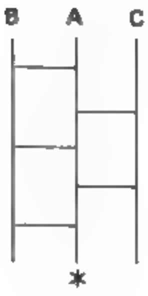
- First, draw an equal number of lines in accordance with the numbers of participants in this Amidakuji lottery, as illustrated at the left.



- Draw a horizontal line between the vertical lines at random. The number of horizontal lines has no limit, and you can draw as many lines as you like. However, when the horizontal line is drawn, this line should be drawn on different levels as illustrated at the left. Dotting a line as shown is not allowed.

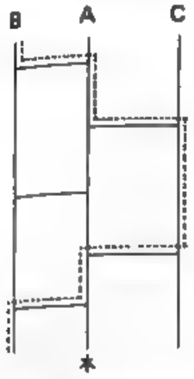


- At the end of the line, the mark " * " is the winning check.
- Next, decide who first draws the line, but the "B" participant cannot select the middle line which "A" participant has first drawn. "B" should select another line to draw, say, either the full end on the left or the right.



- Now, the participant has decided which line he wishes to draw as shown in the left illustration. The draw lines of Amidakuji have been completed. Subsequently, the program of Amidakuji begins.

We shall briefly explain the procedure of how to check the winning mark in the example of participant "B".



- From the top of the line, which participant "B" has selected, to the end of the line, you should trace back as the dotted line shows, when you encounter the horizontal line during tracing from the top, you trace the direction of the line downwards, but never upwards. In the case of participant "B", as the dotted line shows, when traced downwards, "B" cannot reach the marked line. Then, "B" finally loses the lottery in question.

In this procedure, you can play the lottery by repeating it for participant "A" and participant "B" respectively. In the case of the above Amidakuji example, participant "C" reaches the end of the line marked " * ".

■ HOW TO OPERATE

1. R U N ENTER (Program starts)
   In a space with the display on the screen, input the number of persons who take part in AMIDAKUJI and also the number of marks " * " which indicate the winning mark.
2. Next, upon the indication of the display on the screen as "A–>", input the name of the participant who draws the line A. Following the same operation procedure, input the name of all the participants in a regular seguence.
3. When input is finished, AMIDAKUJI will be displayed on the screen. Press the space key to start the line of A at the left side. When the trace reaches the full end of the line from the top, the result either of "win" or "fail" will be displayed. The line of B stands by and is ready to be input for the repetition of tracing from the top to the bottom. Repeat this operation in accordance with the number of participants.
4. When all the numbers of the winning mark " * " are displayed, the program ends.

   Note: The number of the participants is limited from 2 to 10. The number of the winning marks cannot exceed the number of participants.
   The winning mark is permitted from 1 – 5. The name should be input within 10 letters.

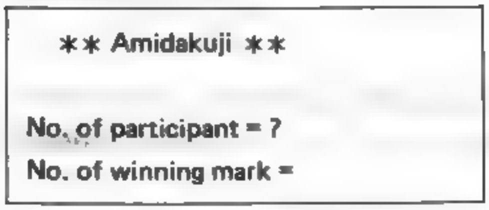
■ EXAMPLE

Participant: 4
Winning mark: 2

Line	Name
A	LUIS
B	BOND
C	ADAMS
D	FORD

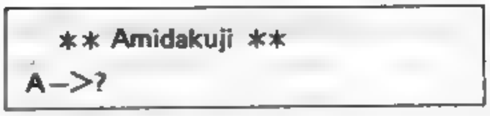
■ KEY OPERATION SEQUENCE

1. [R] [U] [N] [ENTER]
[Program starts]



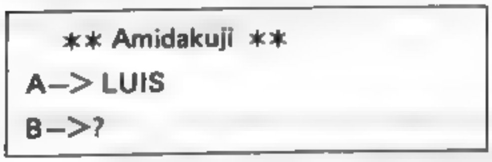
```
  ** Amidakuji **

No. of participant = ?
No. of winning mark =
```

2. 4 [ENTER] 2 [ENTER]
[Input number of participant and number of winning mark]



```
  ** Amidakuji **
A–>?
```

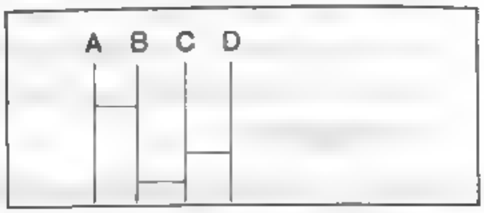
3. [L] [U] [I] [S] [ENTER]
[Input the name of the partcipant who draws line A]



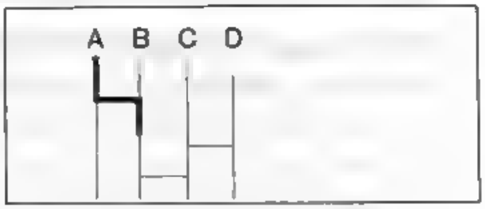
```
  ** Amidakuji **
A–> LUIS
B–>?
```

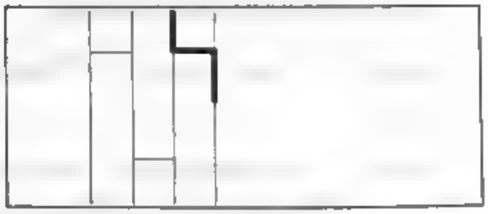
Execute the operation as above, but only in the numbers of the participant.

4. [F] [O] [R] [D] [ENTER]
[Input the name of the participant who draws the line D]

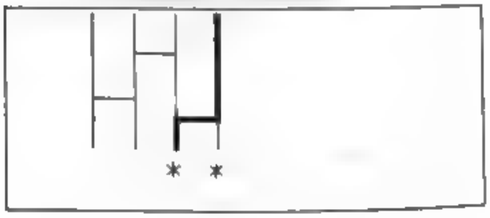


5. □ [A line starts]

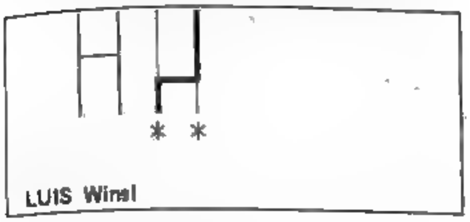




(Trace A line)

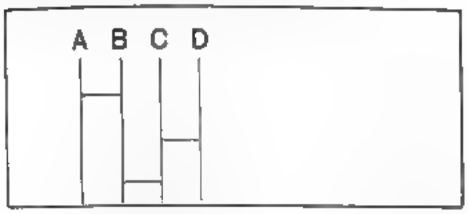


(BEEP Sounds 3 times when the mark is hit)

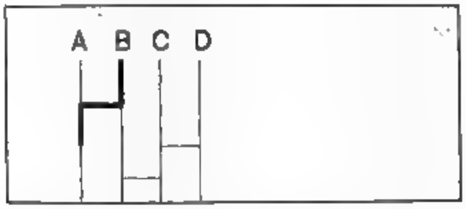


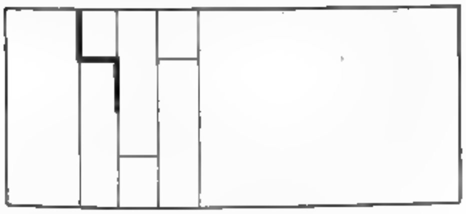
LUIS Win!

(Will display the name of the participant who draws A and then B stands by and proceeds)

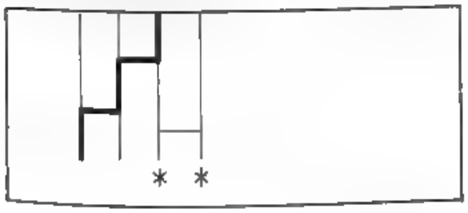


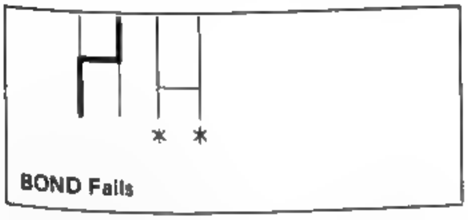
6. ▭ [B line starts]





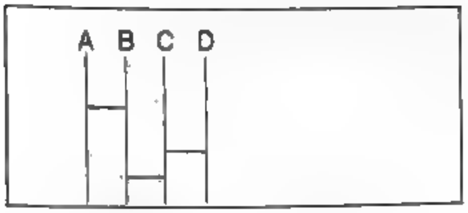
(Trace B line)



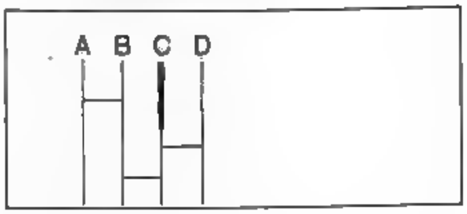


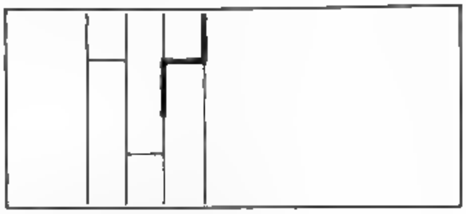
BOND Fails

(Will display the name of the participant who draws line B. C line stands by and proceeds)

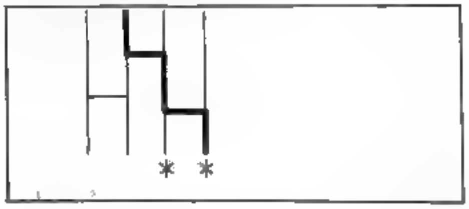


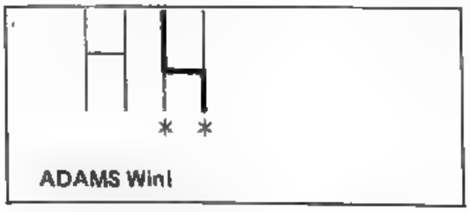
7. ▭ [C line starts]





(Trace C line)





ADAMS Win!

(BEEP sounds 5 times when all winning marks displayed. Program ends.

```
>
```

■ PROGRAM LIST

```
 10:CLS : CLEAR : WAIT 0:H$="          "
 20:PRINT " ** Amidakuji **": BEEP 2:
    PRINT "": PRINT "No.of ParticiPant=
    ": PRINT "No.of winning mark="
 30:CURSOR 18,2: INPUT N:N= INT N
 40:IF N>10 OR N<=1 CURSOR 18,2: PRINT
    "      ": GOTO 30
 50:CURSOR 19,3: INPUT M:M= INT M
 60:IF M>5 OR M>N-1 OR M<=0 CURSOR 19,3
    : PRINT "     ": GOTO 50
 70:BEEP 1: GOSUB 230: GOSUB 170: BEEP
    1: GOSUB 500
 80:FOR P=1 TO N:A=2,Y=8,Z=P*12+14
 90:CLS : WAIT 0: CURSOR 3,0: PRINT C$(
    3): FOR I=1 TO N
100:GCURSOR (I*12+14,15): GPRINT C$(0):
    GCURSOR (I*12+14,23): GPRINT C$(M(1
    ,I)): GCURSOR (I*12+14,31): GPRINT
    C$(M(2,I)): NEXT I
110:IF INKEY$ <>"" THEN 110
120:FOR C=1 TO 8: CLRSOR 23,3: PRINT "
    ": GOSUB 370: PRINT "":A=A+1
130:IF C=8 THEN 150
140:FOR I=1 TO N: GCJRSOR (I*12+14,31):
    GPRINT C$(M(A,I)): NEXT I
150:NEXT C: GOSUB 440: GOSUB 370: WAIT
    150: GOSUB 450: NEXT P: BEEP 5: CLS
160:END
170:FOR I=1 TO N
180:IF R(I)=1 LET W$="* ": GOTO 200
190:W$="  "
200:C$(4)=C$(4)+W$
210:NEXT I
220:RETURN
230:DIM M(10,N),C$(4)*24,O$(N)*22,R(N):
    A=3: RANDOM
240:B=1,X=1: GOSUB 250:B=2,X=2: GOSUB 2
    50: GOTO 290
250:FOR J=X TO N-1 STEP 2
260:FOR I=1 TO 8:R=( RND 2-1)*B:M(I,J)=
    R
270:IF R<>0 LET I=I+1:M(I,J)=0
280:NEXT I: NEXT J: RETURN
290:FOR I=1 TO 9:M(I,N)=0: NEXT I
300:C$(3)= LEFT$ ("A B C D E F G H I J"
    ,N*2):C$(2)="FF040404040404040404
    04"
310:C$(1)="FF2020202020202020202020":C$
    (0)="FF"
320:FOR I=1 TO M
330:R= RND N
340:IF R(R)=1 THEN 330
350:R(R)=1: NEXT I
360:RETURN
370:PSET (Z-1,Y): PSET (Z+1,Y)
380:K= POINT (Z-1,Y+1): IF K PSET (Z+1,
    Y+1): PSET (Z+1,Y+2): GOSUB 430:
    GOTO 410
390:L= POINT (Z+1,Y+1): IF L PSET (Z-1,
    Y+1): PSET (Z-1,Y+2): GOSUB 420
400:IF Y=23 LET Y=16: RETURN
410:Y=Y+1: GOTO 370
420:FOR B=1 TO 13: PSET (Z+B,Y): PSET (
    Z+B,Y+2): NEXT B: PSET (Z+B-1,Y+1):
    Z=Z+B-2,Y=Y+1: RETURN
430:FOR B=1 TO 13: PSET (Z-B,Y): PSET (
    Z-B,Y+2): NEXT B: PSET (Z-B+1,Y+1):
    Z=Z-B+2,Y=Y+1: RETURN
440:CURSOR 3,3: PRINT C$(4): CURSOR 23,
    3: PRINT " ": RETURN
450:IF R((Z-14)/12)=1 BEEP 3: GOTO 470
460:BEEP 1: PRINT O$(P);" Fails": GOTO
    480
470:PRINT O$(P);" Win '":E=E+1
480:IF E=M LET P=N
490:RETURN
500:CLS : PRINT " ** Amidakuji **"
510:IF N>3 LET B=3,C=0: GOTO 530
520:B=N:C=1
530:FOR I=1 TO B: CURSOR : PRINT CHR$ (
    64+I);"->"
540:CURSOR 3,I: INPUT O$(I): IF LEN O$(
    I)>10 CURSOR 3,I: PRINT H$;H$;H$:
    GOTO 540
550:NEXT I
560:IF C RETURN
570:FOR I=4 TO N: FOR J=1 TO 2
580:CURSOR 0,J: PRINT H$;H$;H$: CURSOR
    0,J: PRINT CHR$ (61+I+J);"->";O$(I-
    3+J): NEXT J
590:CURSOR 0,3: PRINT CHR$ (64+I);"->":
    CURSOR 3,3: PRINT H$;H$;H$: CURSOR
    3,3: INPUT O$(I): IF LEN O$(I)>10
    GOTO 590
600:NEXT I: RETURN
```

1596 bytes

■ MEMORY CONTENTS

A	dim. counts of Amidakuji
B	√
C	√
E	counts of executions of Amidakuji
H$	√
I	√
J	√
K	√
L	√
M	number of winning marks
N	participants
P	√
R	random numbers
W$	√
X	√
Y	Y-coordinates of line
Z	Z-coordinates of line
R(N)	√
M(10,N)	√
C$(4)	for display
O$(N)	name

Program Title: **COMPUTER GRAPHIC DESIGNS**

How about drawing computer graphic designs? Computer graphics form figures or patterns with the aid of a computer.

You can enjoy vivid colored computer graphic designs simply by inputting the number of the angles, sizes and the contraction rate of the figures.

■ HOW TO OPERATE

1. Program starts with R U N ENTER .

   In accordance with the display, input the number of angles, sizes and the contraction rate of the figure to allow moving around as well as the instruction of whether to draw the figures around the figure in the center.

   When the figures are not drawn around, select the color of the figures which you output, after inputting your selections from the above data.

2. When the input data has been completed, the output figures will be output to the printer.

   Note: The maximum values for the sizes of the figures feasible for input are as follows.

   - When drawing the figures in and around the center figure .up to 97
   - When not drawing the figures in and around the center figure . . . . .up to 240

■ REFERENCE

- The size of the figures are set by the radius R of the circumscribed circle. 1 unit of R is 0.2 mm.
- The figures move around every 9°.

■ EXAMPLE

1. Number of angle : 5
   Size : 60
   Contraction Rate : 0.9

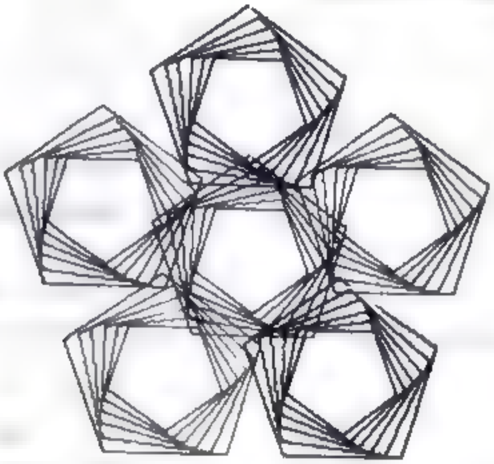
   Draws the figures around the center figure.

2. Number of angle : 6
   Size : 100
   Contraction Rate : 0.9
   Color : Blue

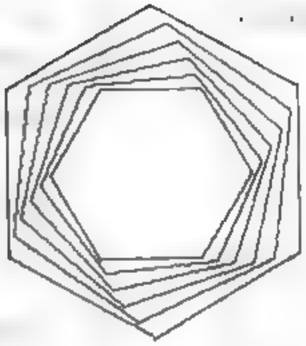
   No figures drawn around the center figure.

■ PRINTOUT (Colored Refer to page 1)

[EXAMPLE 1]



[EXAMPLE 2]



■ KEY OPERATION SEQUENCE

《 Example 1 》

1. R U N ENTER [Program starts at]

```
Angle = ?
Size  =
Reduct. Rate (1 or Ø.9)
Draws in & around (Y/N)
```

2. 5 ENTER 60 ENTER
[Number of angle and size input]

```
Angle = 5
Size  = 6Ø
Reduct. Rate (1 or Ø.9)?
Draws in & around (Y/N)
```

3. 0.9 ENTER [Reduction rate input]

```
Angle = 5
Size  = 6Ø
Reduct. Rate (1 or Ø.9) Ø.9
Draws in & around (Y/N)?
```

4. Y ENTER [Draws the figures in and around]

```
** Computer Graphics **
        Printout
```

```

>
```

(Printout the figures and then program ends.)

《 Example 2 》

1. R U N ENTER [Program starts at]

```
Angle = ?
Size  =
Reduct. Rate = (1 or Ø.9)
Draws in & around (Y/N)
```

2. 6 ENTER 100 ENTER
[Number of angle and size input]

```
Angle = 6
Size  = 1ØØ
Reduct. Rate (1 or Ø.9)?
Draws in & around (Y/N)
```

3. 0.9 ENTER [Reduction rate input]

```
Angle = 6
Size  = 1ØØ
Reduct. Rate (1 or Ø.9) Ø.9
Draws in & around (Y/N)?
```

4. N ENTER [Not draw the figures in and around]

```
Select color?

Black ==>Ø     Green ==>2
Blue  ==>1     Red   ==>3
```

5. 1 ENTER [Select blue color]

```
** Computer Graphics **
        Printout
```

```

>
```

(Printout the figures and then program ends.)

■ PROGRAM LIST

```
10:DEGREE : CLEAR : CONSOLE 39: LPRINT
   : CLS
20:LPRINT CHR$ 27;"a"; CHR$ 13: LPRINT
   CHR$ (27);"b": LPRINT "L0"
30:CL=-1: WAIT 0
40:CURSOR 0,0: PRINT "Angle ="
50:CURSOR 0,1: PRINT "Size  ="
60:CURSOR 0,2: PRINT "Reduct. Rate(1or
   0.9)"
70:CURSOR 0,3: PRINT "Draws in & aroun
   d(Y/N)"
80:CURSOR 8,0: INPUT K: IF K<=2 CLS :
   GOTO 40
90:TH=360/K
100:CURSOR 8,1: INPUT S
110:CURSOR 20,2: INPUT RT
120:IF RT=1 LET JJ=10
130:IF RT=.9 LET JJ=6
140:IF RT<>1 AND RT<>.9 GOTO 110
150:CURSOR 22,3: INPUT Y$
160:IF Y$="Y" OR Y$="y" GOSUB 620: GOTO
    300
170:IF Y$="N" OR Y$="n" GOTO 190
180:GOTO 150
190:SX=240:SY=-S
200:CLS : CURSOR 0,0: PRINT "Select col
    or"
210:CURSOR 0,2: PRINT "Black==>0"
220:CURSOR 0,3: PRINT "Blue ==>1"
230:CURSOR 11,2: PRINT "Green==>2"
240:CURSOR 11,3: PRINT "Red  ==>3"
250:CURSOR 12,0: INPUT CL
260:IF (CL=0)+(CL=1)+(CL=2)+(CL=3)<>1
    GOTO 200
270:CL=CL-1
280:CLS : CURSOR 0,0: PRINT "** Compute
    r Graphics **"
290:CURSOR 7,1: PRINT "Printout": GOSUB
    330
300:LPRINT "R";-KX;",";-KY-3*S: LPRINT
    "A"
310:LPRINT CHR$ 27;"0": CLS
320:END
330:REM  * Write *
340:LPRINT "M";SX;",";SY: LPRINT "I"
350:R=S:KX=0:KY=S
360:IF L=K AND CL=3 LET CL=1
370:IF CL=3 LET CL=0
380:CL=CL+1
390:FOR J=1 TO JJ
400:IF J=1 LET X1=0:Y1=R: GOTO 440
410:R=R*RT
420:X1= INT (R* SIN ((J-1)*(TH/10)))
430:Y1= INT (R* COS ((J-1)*(TH/10)))
440:LPRINT "M";X1;",";Y1
450:FOR I=1 TO K
460:KX= INT (R* SIN (I*TH+(J-1)*(TH/10)
    ))
470:KY= INT (R* COS (I*TH+(J-1)*(TH/10)
    ))
480:LPRINT CHR$ (27);CL
490:LPRINT "D";KX;",";KY
500:X1=KX:Y1=KY
510:NEXT I
520:NEXT J
530:RETURN
540:REM  * Rotation *
550:FOR L=1 TO K
560:SX= INT (S*1.5* SIN (L*TH))
570:SY= INT (S*1.5* COS (L*TH))
580:GOSUB 330
590:LPRINT "M";-SX;",";-SY: LPRINT "I"
600:NEXT L
610:RETURN
620:CLS : CURSOR 0,0: PRINT "** Compute
    r Graphics **"
630:CURSOR 7,1: PRINT "Printout"
640:SX=240:SY=-3*S: GOSUB 330: GOSUB 54
    0
650:RETURN
```

1203 bytes

■ MEMORY CONTENTS

I	√
J	√
K	number of angles
L	√
R	semidiameter of external contact circle
S	size
Y$	√
CL	color
JJ	√
RT	reduction rate
TH	angle
SX	X-coordinates on original point
SY	Y-coordinates on original point
X1	X-coordinates before turning
Y1	Y-coordinates before turning
KX	X-coordinates after turning
KY	Y-coordinates after turning

Program Title: **BIORHYTHM**

How is your physical condition this month?

Physical, Sensitivity and Intellectual have their own independent set of cycles. Biorhythms are based on the theory that each of the cycles repeats self with various, good or bad condition within the period of each cycle. This program makes this theory available for drawing up the curved lines of the biorhythm.

■ HOW TO OPERATE

1. R U N ENTER (Program starts.)
   Input your name and date of birth as well as the desired month and year in regular seguence in accordance with the display.
2. Upon input, the curved lines of the biorhythm shall be relayed to the printer in different colors depending on each element such as physical (blue), sensitivity (red) and intellectual (green).

   Note: Input your name within the limit of 10 letters.

■ REFERENCE

Calculation for the X-coordinate values of the curves is done as follows:

Physical	X = Sin ((B + Y)/23 X 360) X 150
Sensitivity	X = Sin ((C + Y)/28 X 360) X 150
Intellectual	X = Sin ((D + Y)/33 X 360) X 150

B, C, and D represent the remainder after the total number of days from the birthday to the desired time has been divided by the individual cycles.

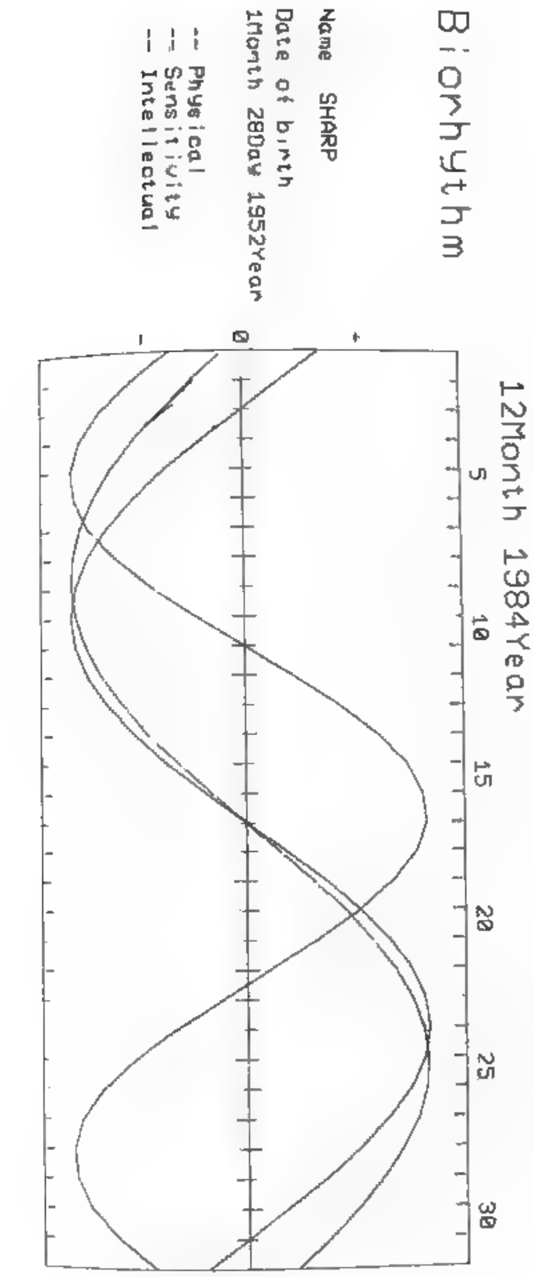
Y represents the number of days.
The maximum length is 16 mm in the positive (+) and negative (—) directions.

Cycle:	Physical:	23 days
	Sensitivity:	28 days
	Intellectual:	33 days

■ EXAMPLE

Type in the following:
Desired month: 12 (December), 1984
Name: SHARP
Date of birth: 1 (January) 28, 1952

PRINTOUT (Colored Refer to page 1)



Biorhythm

Name SHARP

Date of birth

1Month 28Day 1952Year

-- Physical

-- Sensitivity

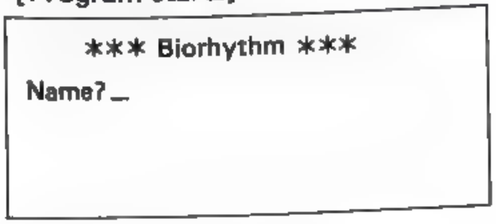
-- Intellectual

12Month 1984Year

KEY OPERATION SEQUENCE

1. R U N ENTER

[Program starts]

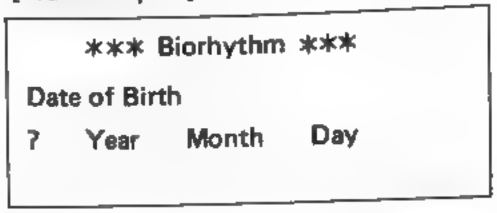


*** Biorhythm ***

Name? _

2. S H A R P ENTER

[Name input]

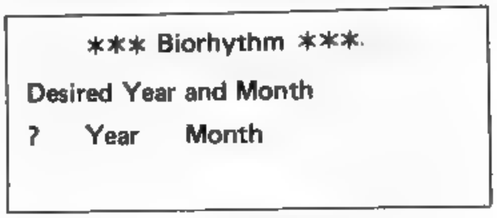


*** Biorhythm ***

Date of Birth

? Year Month Day

3. 1952 ENTER 1 ENTER 28 ENTER



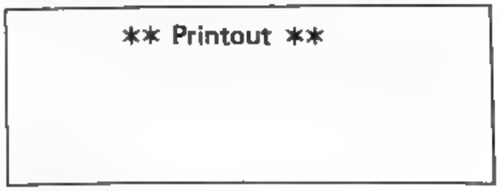
*** Biorhythm ***

Desired Year and Month

? Year Month

4. 1984 ENTER 12 ENTER

[Desired year and month input]



** Printout **

>

(The biorhythm curved line is relayed to the printer and the program ends.)

■ PROGRAM LIST

```
 10:CLEAR : DEGREE : CLS : CONSOLE 39:
    LPRINT : WAIT 0: DIM A$(0)*22,H$(0)
    *25
 20:LPRINT CHR$ 27;"a": LPRINT CHR$ 13:
    LPRINT CHR$ 27;"b": LPRINT CHR$ 27;
    "0": LPRINT "L0"
 30:H$(0)="   *** Biorhythm ***"
 40:PRINT H$(0): INPUT "Name?";A$(0):
    IF LEN A$(0)>10 CLS : GOTO 40
 50:CLS : PRINT H$(0): PRINT "Date of B
    irth": CURSOR 5,2
 60:PRINT "Year": CURSOR 13,2: PRINT "M
    onth": CURSOR 21,2: PRINT "Day"
 70:CURSOR 0,2: INPUT S$:G= VAL S$
 80:IF S$="" CURSOR 0,2: PRINT "
    GOTO 70
 90:CURSOR 10,2: INPUT T
100:IF T<1 OR T>12 GOTO 90
110:CURSOR 18,2: INPUT U
120:IF U<1 OR U>31 GOTO 110
130:S=G
140:CLS : PRINT H$(0): PRINT "Desired Y
    ear and Month": CURSOR 5,2
150:PRINT "Year": CURSOR 13,2: PRINT "M
    onth"
160:CURSOR 0,2: INPUT L$:G= VAL L$
170:IF L$="" CURSOR 0,2: PRINT "     ":
    GOTO 160
180:CURSOR 10,2: INPUT M
190:IF M<1 OR M>31 GOTO 180
200:CLS : CURSOR 2,1: PRINT "** Printou
    t **"
210:LPRINT "M400,-35": LPRINT "Q1":
    LPRINT CHR$ 27;"?d"
220:LPRINT "PBiorhythm": LPRINT CHR$ 27
    ;"?b"
230:LPRINT "M300,-35": LPRINT "PName  ";
    A$(0): LPRINT "M265,-35"
240:LPRINT "PDate of Birth"
250:LPRINT "M240,-35": LPRINT "P"; STR$
    T;" Month "; STR$ U;" Day "; STR$ S;"
    Year"
260:LPRINT "M190,-35": LPRINT CHR$ 27;"
    1": LPRINT "P -- Physical"
270:LPRINT "M170,-35": LPRINT CHR$ 27;"
    3": LPRINT "P -- Sensitivity"
280:LPRINT "M150,-35": LPRINT CHR$ 27;"
    2": LPRINT "P -- Intellectual"
290:LPRINT CHR$ 27;"0"
300:LPRINT CHR$ 27;"?c": LPRINT "M448,-
    350": LPRINT "P"; STR$ M;"Month ";
    STR$ G;"Year"
310:L=G: GOSUB 790: GOSUB 740
320:O=A,L=S,M=T,N=U: GOSUB 740:P=A
330:A=O-P
340:LPRINT CHR$ 27;"?b"
350:LPRINT "M230,-325": LPRINT "I"
360:Y=I*-25
370:LPRINT "M180,0": LPRINT "J-360,0"
380:LPRINT "J0,";Y;",360,0,0,";-Y
390:LPRINT "H": LPRINT "D0,";Y
400:LPRINT "M90,15": LPRINT "P+":
    LPRINT "M-5,15": LPRINT "P0":
    LPRINT "M-90,15"
410:LPRINT "P-"
420:G=1: FOR Q=1 TO I-1
430:Y=Q*-25
440:ON G GOTO 450,470
450:LPRINT "M-180,";Y: LPRINT "J10,0":
    LPRINT "R160,0"
460:LPRINT "J20,0": LPRINT "R160,0":
    LPRINT "J10,0":G=2: GOTO 490
470:LPRINT "J-10,0": LPRINT "R-160,0":
    LPRINT "J-20,0"
480:LPRINT "R-160,0": LPRINT "J-10,0":G
    =1
490:LPRINT "R0,-25"
500:NEXT Q
510:FOR Q=5 TO I STEP 5
520:Y=Q*-25+25
530:LPRINT "M190,";Y: LPRINT "P"; STR$
    Q
540:NEXT Q
550:B= INT (A/23):B=A-(23*B)
560:C= INT (A/28):C=A-(28*C)
570:D= INT (A/33):D=A-(33*D)
580:FOR J=1 TO 3
590:LPRINT CHR$ 27;J
600:FOR Y=1 TO I+1
610:ON J GOSUB 710,730,720
620:Z=Y*-25+25
630:IF Y=1 LPRINT "M";X;",";Z
640:LPRINT "D";X;",";Z
650:NEXT Y
660:NEXT J
670:LPRINT "R-230,-200"
680:LPRINT CHR$ 27;"0"
690:LPRINT CHR$ 27;"a"
700:CLS : END
710:X= SIN ((B+Y)/23*360)*150: RETURN
720:X= SIN ((C+Y)/28*360)*150: RETURN
730:X= SIN ((D+Y)/33*360)*150: RETURN
740:IF M-3>=0 LET M=M+1: GOTO 780
```

```
750:L=L-1:M=13+M
760:A= INT (365.25*L)+ INT (30.6*M)+N
770:A=A- INT (L/100)+ INT (L/400)
780:RETURN
790:IF M=2 GOTO 860
800:IF M=4 GOTO 850
810:IF M=6 GOTO 850
820:IF M=9 GOTO 850
830:IF M=11 GOTO 850
840:I=31: GOTO 960
850:I=30: GOTO 960
860:K= INT (L/4):K=L-K*4
870:IF K=0 GOTO 890
880:I=28: GOTO 960
890:K= INT (L/100):K=L-K*100
900:IF K=0 GOTO 920
910:GOTO 950
920:K= INT (L/400):K=L-K*400
930:IF K=0 GOTO 950
940:GOTO 880
950:I=29
960:RETURN
```

1989 bytes

■ MEMORY CONTENTS

A	the total number of days
B	the total number of days/remaining number of the cycle
C	the total number of days/remaining number of the cycle
D	the total number of days/remaining number of the cycle
G	√
I	number of days of desired month
J	loop counter
K	√
L, L$	desired year, √
M	desired month, √
N	√
O	√
P	√
Q	√
S, S$	√, √
T	month of birth
U	date of birth
X	X-coordinates on biorhythm curved line
Y	counter of the days of one month
Z	Y-coordinates on biorhythm curved line
A$(Ø)	name
H$(Ø)	for storing the titles

Program Title:	**ABC ANALYSIS**	RAM CARD (CE-201M) required

This program ranks the products. Persons in charge of sales should hold a firm grasp of what position each product occupies in the general sales returns.

The program for which the existing ABC analysis of the stock control method has been applied will calculate the ratio of the sales amount and the ratio of the aggregate total when the identification number of the products and the sales returns are input. This will print out the above ratio and will carry out ABC ranking as well.

▪ HOW TO OPERATE

1. R U N ENTER (Program starts.)
   Input the identification number of each product and sales amount. While the identification number of the product stands by for input, you can shift to the control 2 if you operate ENTER only.

2. Next, select whether or not you should amend the data you have input. When you select the data, operate Y ENTER , and then if the data is incorrect, input the correct data, because the sales returns and the identification number of the products are displayed just by operating ENTER only. When you do not need to amend the data, operate N ENTER .

3. Output the sales amount ratio, the aggregate total ratio and ABC ranking as well. When you output their results in the printer, operate P ENTER . When you display the results, operate L ENTER . When you cannot display the whole figures and letters at a time, press ↑ (up) and ↓ (down) to read the hidden portions of the whole. Operate E ENTER and then the process of the program is completed.

4. When you output the results into the printer or display them on the screen, you can shift again to the control 3.

   Note: The display of the identification number of the products has a maximum of up to 7 letters and that of the sales returns up to 6 figures. The items you can input have a maximum of up to 41 different items only.
   This is due to the limited capacity of the values to read from each applied command when the graph is printed into the built-in printer.

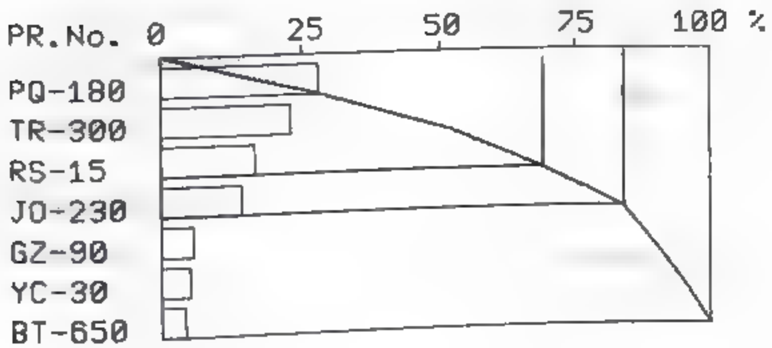
■ EXAMPLE

The sales amount ratio and the aggregate ratio are calculated from the data below and ABC ranking is executed.

Product No.	Sales amount
TR–300	89,000
BT–650	17,000
GZ–90	23,000
RS–15	65,000
JO–230	56,000
PQ–180	108,000
YC–30	21,000

■ PRINTOUT (Colored Refer to page 2)

```
                 ** A B C Analysis **

PR. No.   Sales    %      Sum     %    Rank
---------------------------------------------
PQ-180   108000 28.4   108000   28.4   (A)
TR-300    89000 23.4   197000   51.9   (A)
RS-15     65000 17.1   262000   69.1   (A)
JO-230    56000 14.7   318000   83.9   (B)
GZ-90     23000  6.0   341000   89.9   (C)
YC-30     21000  5.5   362000   95.5   (C)
BT-650    17000  4.4   379000  100.0   (C)
---------------------------------------------
```



■ KEY OPERATION SEQUENCE

1. R U N ENTER
   [Program starts]

```
(1) Product No.     Sales
     ?
```

2. TR–300 ENTER [1st product No. input]

```
(1) Product No.     Sales
     TR–300        ?
```

3. 88000 ENTER
   [1st product sales returns input]

```
(1) Product No.     Sales
     TR–300        88000
(2) Product No.     Sales
     ?
```

:
Input same as above
:

4. YC–30 ENTER
   [7th product number input]

```
(6) Product No.     Sales
     PQ–180        108000
(7) Product No.     Sales
     YC–30         ?
```

5. 21000 ENTER
   [7th product sales returns input]

```
(7) Product No.     Sales
     YC–30         21000
(8) Product No.     Sales
     ?
```

6. ENTER [Select data confirmation?]

```
      ** A B C  Analysis **

Confirm or Amend (Y/N)_
```

7. Y ENTER [Confirm data]

```
(1) TR–300–> ?
```

8. ENTER

```
(1) TR–300 –>?
     88000 –>?
```

9. 89000 ENTER [Correct data input]

```
(1) TR–300 –>?
     88000 –>89000
(2) BT–650 –>?
```

:
Input same as above
:

10. ENTER

```
(6) PQ–180 –>?
     108000  –>?
(7) YC–30 –>?
     21000  –>?
```

11. ENTER
    [Select reconfirmation of data?]

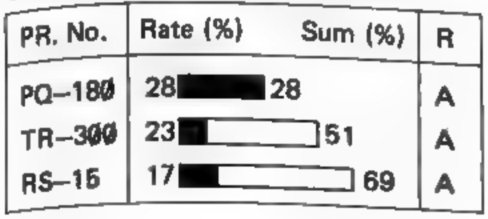
```
      ** A B C  Analysis **

   Confirm or Amend (Y/N)_
```

12. N ENTER
    [Unconfirmed data]

```
      ** A B C  Analysis **

     LCD or PRINTER or END
              (L/P/E)?_
```

13. L ENTER

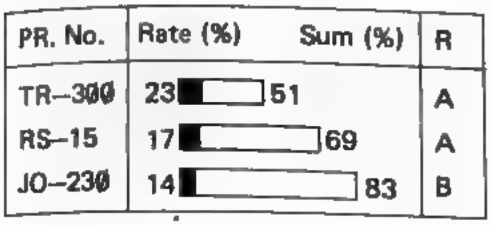
[Output method select display]



```
PR. No.  | Rate (%)      Sum (%) | R
PQ-18Ø   | 28█████████ 28        | A
TR-3ØØ   | 23█▭▭▭▭▭▭▭ 51         | A
RS-15    | 17██▭▭▭▭▭▭▭▭ 69       | A
```

14. ↓

[Display confirmation by scroll down]



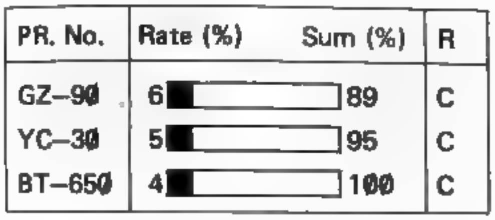
```
PR. No.  | Rate (%)      Sum (%) | R
TR-3ØØ   | 23█▭▭▭▭ 51            | A
RS-15    | 17█▭▭▭▭▭▭ 69          | A
JO-23Ø   | 14█▭▭▭▭▭▭▭▭ 83        | B
```

⋮

Input same as above

⋮

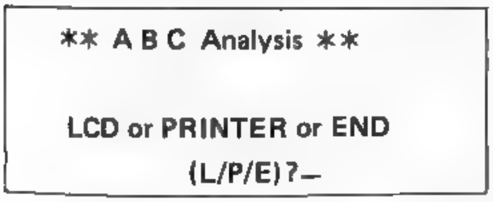
15. ↓



```
PR. No.  | Rate (%)      Sum (%) | R
GZ-9Ø    | 6█▭▭▭▭▭▭▭▭ 89         | C
YC-3Ø    | 5█▭▭▭▭▭▭▭▭ 95         | C
BT-65Ø   | 4█▭▭▭▭▭▭▭▭ 1ØØ        | C
```

16. Q

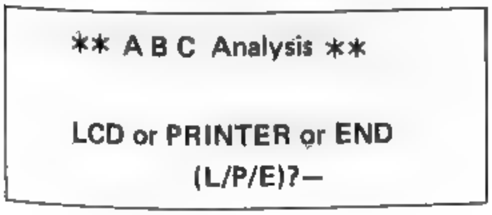
[Select output method repeat]



```
   ** A B C Analysis **

   LCD or PRINTER or END
         (L/P/E)?_
```

17. P ENTER

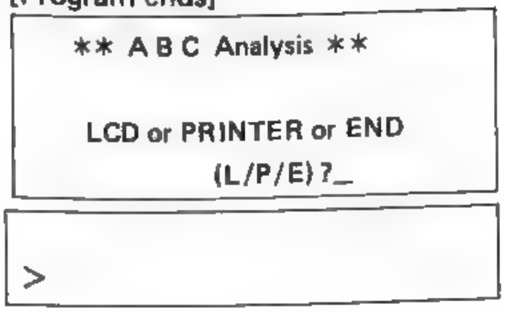
[Output method select print]



```
   ** A B C Analysis **

   LCD or PRINTER or END
         (L/P/E)?_
```

18. E ENTER

[Program ends]



```
   ** A B C Analysis **

   LCD or PRINTER or END
         (L/P/E)?_

>
```

■ PROGRAM LIST

```
 10:CLEAR : CLS : CONSOLE 39: LPRINT :
    WAIT 0:A=41:B=14: DIM B$(A)*B,C(A),
    D(A),K$(A)*1,Z$(0)*6
 20:FOR I=1 TO A: CURSOR : IF I>2 GOSUB
    660: PRINT " "
 30:CURSOR 0,(I>1)*2: PRINT "("; STR$ I
    ;")Product No.  Sales": PRINT " "
 40:BEEP 1: CURSOR 2+ LEN STR$ I,(I=1)+
    (I>1)*3: INPUT B$(I)
 50:IF B$(I)="" LET C=I-1:I=A: GOTO 120
 60:B$(I)= LEFT$ (B$(I),7)
 70:CURSOR 2+ LEN STR$ I,(I=1)+(I>1)*3:
    PRINT B$(I);"              ";
 80:CURSOR 14+ LEN STR$ I,(I=1)+(I>1)*3
    : BEEP 1: INPUT Z$(0)
 90:CURSOR 14+ LEN STR$ I,(I=1)+(I>1)*3
    : PRINT Z$(0);'   ";
100:C(I)= VAL Z$(0): IF C(I)<=0 BEEP 1:
    GOTO 80
110:E=E+C(I):C=I:Z$(0)="'
120:NEXT I: IF C<2 BEEP 2: GOTO 10
130:BEEP 1: CLS : CURSOR 1,0: PRINT "*
    * A B C Analysis * *"
140:CURSOR 0,2: INPUT 'Confirm or Amend
     (Y/N)';Z$
150:IF Z$='N' OR Z$='n' GOSUB 490: GOTO
    260
160:IF Z$='Y' OR Z$="y" THEN 180
170:GOTO 130
180:CLS : FOR I=1 TO C
190:CURSOR : IF I>2 GOSUB 660: PRINT "

200:CURSOR 0,(I>1)*2: PRINT "("; STR$ I
    ;")";B$(I);" -> ": CURSOR 6+ LEN
    STR$ I+ LEN B$(I),(I>1)*2
210:INPUT B$(I):B$(I)= LEFT$ (B$(I),7)
220:CURSOR 2+ LEN STR$ I,(I=1)+(I>1)*3:
    PRINT STR$ C(I);" -> ": CURSOR LEN
    STR$ C(I)+7,(I=1)+(I>1)*3
230:Z$(0)= STR$ C(I)
240:INPUT Z$(0): IF VAL Z$(0)<=0 BEEP 1
    : GOTO 220
250:E=E-C(I):C(I)= VAL Z$(0):E=E+C(I):
    NEXT I: CLS : GOTO 130
260:CLS :H=0:V=0: CURSOR 1,0: PRINT "*
    * A B C Analysis * *"
270:CURSOR 1,2: PRINT "LCD or PRINTER o
    r END"
280:BEEP 1: CURSOR 8,3: INPUT "(L/P/E)?
    ";Z$
290:IF Z$="E" OR Z$="e" BEEP 3: CLS :
    END
300:IF Z$="P" OR Z$="p" THEN 670
310:IF Z$="L" OR Z$="l" THEN 330
320:GOTO 260
330:CLS : CURSOR 0,0: PRINT "PR. No. Ra
    te(%)": CURSOR 16,0: PRINT " Sum(%)
    R":Y=1
340:FOR I=1 TO 3-(C=2): CURSOR 0,I:
    GOSUB 660:V=D(Y-1)
350:CURSOR 0,I: PRINT B$(Y): CURSOR 8,I
    :D$= STR$ ( INT (D(Y)-V))
360:IF LEN O$=1 LET O$=" "+D$
370:PRINT O$
380:CURSOR (64+D(Y)/2)/6-.9,I: PRINT
    USING "####";D(Y); CURSOR 23,I:
    PRINT K$(Y)
390:LINE (66,I*8+1)-(66+(D(Y)-V)/2,I*8+
    5),BF
400:LINE (66+(D(Y)-V)/2,I*8+1)-(66+D(Y)
    /2,I*8+5),B
410:Y=Y+1: NEXT I
420:LINE (2,7)-(149,7): LINE (2,31)-(14
    9,31): LINE (2,0)-(2,31)
430:LINE (48,0)-(48,31): LINE (143,0)-(
    143,31): LINE (149,0)-(149,31)
440:IF INKEY$ ="" GOTO 440
450:IF INKEY$ ="Q" OR INKEY$ ="q" GOTO
    260
460:IF INKEY$ = CHR$ (31) AND Y<C+1
    BEEP 1:Y=Y-2: GOTO 340
470:IF INKEY$ <> CHR$ (30) OR Y<5 BEEP
    2: GOTO 440
480:Y=Y-4: BEEP 1: GOTO 340
490:L= INT (C/2+1):R=C
500:IF L>1 LET L=L-1:X=C(L):B$(0)=B$(L)
    : GOTO 530
510:X=C(R):B$(0)=B$(R):C(R)=C(1):B$(R)=
    B$(1):R=R-1
520:IF R<=1 LET C(1)=X:B$(1)=B$(0):
    GOTO 600
530:J=L
540:I=J:J=2*J
550:IF J>R THEN 590
560:IF J<R THEN IF C(J)>C(J+1) LET J=J+
    1
570:IF X<=C(J) THEN 590
580:C(I)=C(J):B$(I)=B$(J): GOTO 540
590:C(I)=X:B$(I)=B$(0): GOTO 500
600:FOR I=1 TO C
610:H=H+C(I):D(I)=H/E*100
620:IF D(I)<=70 OR I=1 LET K$(I)="A":
    GOTO 650
630:IF D(I)<=85 LET K$(I)="B": GOTO 650
640:K$(I)="C"
650:NEXT I: RETURN
660:PRINT "                        :
    RETURN
670:CLS : CURSOR 5,1: PRINT "** Printou
    t **"
680:LPRINT CHR$ 13: LPRINT CHR$ 27;"a";
    CHR$ 27;"0"; CHR$ 27;"?b'
690:LPRINT "            ** A B C Analysis
     **  ": LPRINT
700:LPRINT "PR. No.  Sales   %     Sum
       %   Rank"
```

```
710:LPRINT "----------------------------------------
    -------------"
720:FOR I=1 TO C:H=H+C(I)
730:LPRINT USING "&&&&&&&";B$(I);
740:LPRINT USING "#######";C(I); USING
    "###.#";D(I)-V; USING "#########";H;
750:LPRINT USING "####.#";D(I); USING ;
    "  (";K$(I);")":V=D(I): NEXT I
760:LPRINT "----------------------------------------
    -------------": LPRINT : LPRINT CHR$
    15
770:H=0:V=0
780:LPRINT "PR.No. 0        25       50
     75    100 %"
790:LPRINT CHR$ 27;"b";"R90,15": LPRINT
    "I": LPRINT "L0"
800:LPRINT "D";340;",";0;",";340;",";C*
    -24;",";0;",";C*-24;",";0;",";0
810:FOR I=1 TO 3: LPRINT "R";25*3.4;",
    ;0: LPRINT "J";0;",";5;",";0;",";-5
    : NEXT I
820:LPRINT "H"
830:LPRINT "R";0;",";-6
840:FOR I=1 TO C: LPRINT "J";(D(I)-V)*3
    .4;",";0;",";0;",";-10;",";(V-D(I))
    *3.4;",";0
850:LPRINT "R";0;",";-6:V=D(I): NEXT I
860:LPRINT "H": LPRINT CHR$ 27;"2"
870:FOR I=1 TO C: LPRINT "D";D(I)*3.4;"
    ,";I*-24: NEXT I
880:LPRINT CHR$ 27;"3"
890:FOR I=1 TO C-1: IF K$(I)<>K$(I+1)
    THEN GOSUB 920
900:NEXT I
910:GOTO 940
920:LPRINT "M";D(I)*3.4;",";0: LPRINT "
    D";D(I)*3.4;",";I*-24;",";0;",";I*-
    24
930:RETURN
940:LPRINT "H": LPRINT CHR$ 27;"0":
    LPRINT CHR$ 27;"a"
950:FOR I=1 TO C: LPRINT USING "&&&&&&&
    ";B$(I): NEXT I
960:LPRINT : LPRINT : USING : GOTO 260
```

2889 bytes

■ MEMORY CONTENTS

A	maximum number of data
B	letter length of product No.
C	number of data
E	general sales returns
H	√
I	loop counter, pointer
J	pointer
L	√
O$	√
R	end pointer
V	√
X	√
Y	√
Z$	√
C(A)	sales returns
D(A)	total ratio
B$(A)	product No.
K$(A)	rank
Z$(0)	√

■ KEY OPERATION SEQUENCE

1. [R] [U] [N] [ENTER]
   [Program starts]

```
    ** B.E.P. Analysis **
Sales U.P. =>_
```

2. 75 [ENTER]
   [Sales Unit Price input]

```
    ** B.E.P. Analysis **
Sales U.P. => 75
Vari. U.P. => _
```

Input same as above

3. 5500 [ENTER]
   [Fixed Cost input]

```
Vari. U.P. => 42.5
Sales Q'ty => 22ØØ
Fixed Cost => 55ØØ
Correction? (Y/N) => _
```

4. [Y] [ENTER] [Data modification input]

```
1: Sales U/P   2: Vari.  U/P
3: Sales Q'ty  4: F.C.   5: End
Number => _
```

5. [1] [ENTER] [Input identification No. of item of modified data]

```
1: Sales U/P   2: Vari.  U/P
3: Sales Q'ty  4: F.C.   5: End
Number => 1
Input value => _
```

6. 74 [ENTER] [Correct data input]

```
1: Sales U/P   2: Vari.  U/P
3: Sales Q'ty  4: F.C.  5: End
Number => _
```

7. 5 [ENTER] [No modification of data]

```
    ** B.E.P. Analysis **


        ** Printout **
```

(Graph Analysis of B.E.P. will be displayed.)

```
Simulation? (Y/N) => _
```

8. [Y] [ENTER] [Simulation]

```
1: Sales U/P   2: Vari.  U/P
3: Sales Q'ty  4: F.C.   5: End
Number => _
```

9. 1 [ENTER]

```
1: Sales U/P   2: Vari.  U/P
2: Sales Q'ty  4: F.C.   5: End
Number => 1
Input value  => _
```

10. 77.5 [ENTER]

```
1: Sales U/P   2: Vari.  U/P
3: Sales Q'ty  4: F.C.   5: End
Number => _
```

11. 5 [ENTER]

```
    ** B.E.P. Analysis **


        ** Printout **
```

(Simulation results will be displayed. Program ends.)

```

>
```

Program Title: **ANALYSIS OF BREAK-EVENPOINT OF PROFIT/LOSS** — RAM CARD (CE-201M) required

Steady and sound management with an accurate forecast of company profit!

In respect to the profit forecast for the company in a long-range plan, the ultimate purpose of the company's activities in these days is to pursue the break-evenpoint of profit/loss.
A graph and table analysis of break-evenpoint is obtainable in this software program by means of inputting each available piece of data.

▪ HOW TO OPERATE

1. R U N ENTER (Program starts.) Input Unit Price of Sales, Variable Unit Prices, Quantities of the Sales and Fixed Cost in turn according to the indication of the display.

2. Upon the input, determine whether the data should be modified or not.
   - In the case of modification, operate Y ENTER . Then the table of all the input descriptions of items together with the identification number of items will be displayed. Input the identification number which you want to modify and its data. The table of all identification number of items and descriptions of them will be displayed again. Operate 5 ENTER and then the break-evenpoint analysis' graph and table will be output.
   - If no modification is required, operate N ENTER , and then the break-evenpoint analysis' graph and table will be output.

3. After you output the results, decide whether simulation is read.
   - If required, operate Y ENTER . The tables of all identification number and descriptions of items will be displayed. Input the identification numbers of the data which you want to modify and input the revised data. The table of all identification numbers of items will be displayed. Operate 5 ENTER and obtain simulation results. The program ends.
   - If simulation is not required, operate E ENTER , and then the program ends.

   Note: For example, in the case that Sales Amount equals Variable Unit price, " ** Data is inconsistent ** " will be displayed and then a data modification routine will be set.

▪ CONTENTS

The break-evenpoint of profit/loss means the circumstance at which the incoming profit becomes equal to the outgoing expenditures, or the sales amount which does not produce any profit or loss.

$$\text{Break-Evenpoint of Profit/Loss (Amount)} = \frac{\text{Fixed Cost}}{1 - \dfrac{\text{Variable Cost}}{\text{Sales Amount}}}$$

$$\text{Break-Evenpoint of Profit/Loss (Quantity)} = \frac{\text{Break-Evenpoint (Amount)}}{\text{Sales Unit Price}}$$

$$\text{Profit} = \text{Sales Amount} - \text{Variable Cost} - \text{Fixed Cost}$$

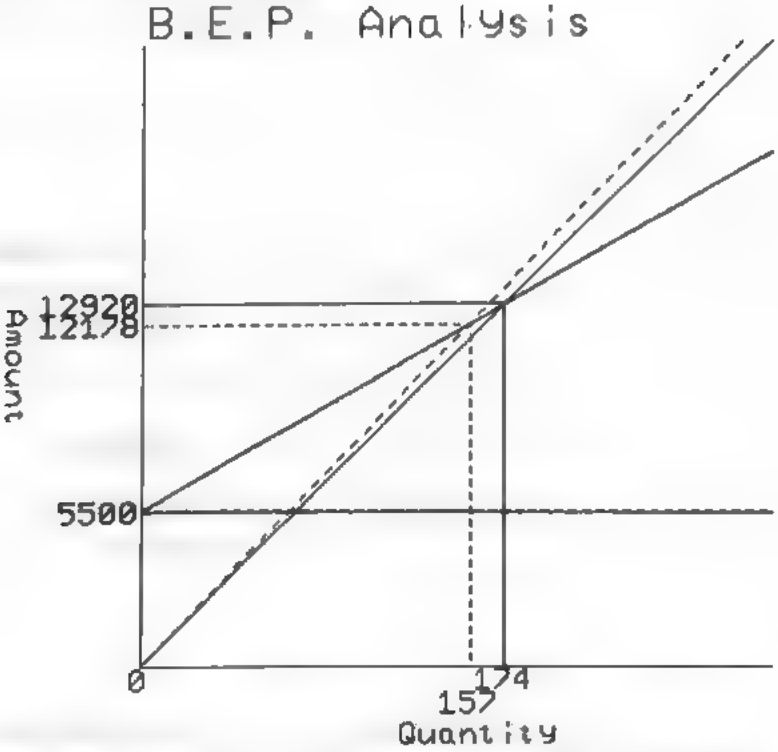
■ EXAMPLE

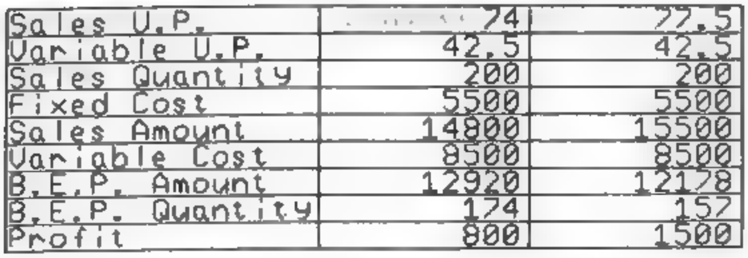
From the data below, complete the analysis graph and analysis table of Break-Evenpoint of Profit/Loss.

Conduct the simulation operation to check how the said graph and table will be displayed when the unit price of the sales is 77.5 modified to . . . . .

Sales Unit Price	74
Variable Unit Price	42.5
Sales Quantity	200
Fixed Cost	5500

■ PRINTOUT (Colored Refer to page 2)





Sales U.P.	74	77.5
Variable U.P.	42.5	42.5
Sales Quantity	200	200
Fixed Cost	5500	5500
Sales Amount	14800	15500
Variable Cost	8500	8500
B.E.P. Amount	12920	12178
B.E.P. Quantity	174	157
Profit	800	1500

■ PROGRAM LIST

```
10:CLEAR : CLS : CONSOLE 39: LPRINT :
   WAIT 0: LPRINT CHR$ 27;"a"; CHR$ 13
20:DIM M$(0)*30
30:REM  * Initialize
40:XW=375:YW=375:L=0:O=0:P=0:Q=0
50:PRINT "**  B.E.P. Analysis  **"
60:INPUT "Sales U.P.=>";A
70:INPUT "Vari. U.P.=>";B
80:INPUT "Sales Q'ty=>";C
90:INPUT "Fixed Cost=>";D
100:INPUT "Correction ?(Y/N) =>";Z$
110:IF Z$="Y" OR Z$="y" THEN 140
120:IF Z$="N" OR Z$="n" THEN 250
130:GOTO 100
140:CLS : CURSOR 0,0: PRINT "1:Sales U/
    P 2:Vari. U/P "
150:CURSOR 0,1: PRINT "3:Sales Q'ty4:F.
    C. 5:End"
160:CURSOR 0,2: INPUT "Number=>";J
170:IF J<1 OR J>5 THEN 160
180:IF J=5 THEN 250
190:CURSOR 0,3: INPUT "Input value=>";S
200:ON J GOTO 210,220,230,240
210:A=S: GOTO 140
220:B=S: GOTO 140
230:C=S: GOTO 140
240:D=S: GOTO 140
250:IF A<>B THEN 270
260:CLS : WAIT 100: CURSOR 1,1: PRINT "
    Data is inconsisent": WAIT 0: GOTO
    140
270:CLS : CURSOR 0,0: PRINT "**  B.E.P.
     Analysis  **"
280:CURSOR 5,2: PRINT "** Printout **"
290:E=A*C:F=B*C:G= INT (D/(1-(F/E))):H=
    INT (G/A):K=E-F-D
300:IF L<>0 THEN 650
310:MX= INT (C*1.5): IF H>C THEN LET MX
    = INT (H*1.5)
320:MY= INT (E*1.5): IF G>E THEN LET MY
    = INT (G*1.5)
330:FX=XW/MX:FY=YW/MY
340:REM  * Frame Print
350:LPRINT CHR$ 27;"0": LPRINT CHR$ 27;
    "b"
360:LPRINT CHR$ 27;"?c": LPRINT "M";85;
    ",";-20: LPRINT "L";0
370:LPRINT "P";"B.E.P. Analysis":
    LPRINT CHR$ 27;"?b"
380:LPRINT "M";80;",";-400: LPRINT "I"
390:REM  * X-Axis
400:LPRINT "J";XW;",";0
410:LPRINT "M";155;",";-45: LPRINT "P";
    "Quantity"
420:REM  * Y-Axis
430:LPRINT "M";0;",";0: LPRINT "J";0;",
    ";YW
440:LPRINT "M";-80;",";215: LPRINT "Q1"
450:LPRINT "P";"Amount": LPRINT "Q0"
460:LPRINT "M";-6;",";-14: LPRINT "P0"
470:REM  * Frame of table
480:SX=-80:EX=355:SY=-55:EY=-199:SW=1:Y
    =-55
490:FOR I=1 TO 10
500:IF SW=1 THEN LPRINT "M";SX;",";Y:
    LPRINT "D";EX;",";Y
510:IF SW<>1 THEN LPRINT "M";EX;",";Y:
    LPRINT "D";SX;",";Y
520:Y=Y-16:SW=-SW
530:NEXT I
540:LPRINT "M";SX;",";SY: LPRINT "D";SX
    ;",";EY
550:LPRINT "M";SX+37/.2;",";EY: LPRINT
    "D";SX+37/.2;',";SY
560:LPRINT "M";SX+62/.2;",";SY: LPRINT
    "D";SX+62/.2;',";EY
570:LPRINT "M";EX;",";EY: LPRINT "D";EX
    ;",";SY
580:RESTORE 620:X=-77:Y=-53
590:FOR I=1 TO 9:Y=Y-16: READ M$(0)
600:LPRINT "M";X;",";Y: LPRINT "P";M$(0
    )
610:NEXT I
620:DATA "Sales U.P.","Variable U.P.","
    Sales Quantity"
630:DATA "Fixed Cost","Sales Amount","V
    ariable Cost"
640:DATA "B.E.P. Amount","B.E.P. Quanti
    ty","Profit"
650:LPRINT "H"
660:LPRINT "L";L
670:M=(E*FY)/(C*FX):N=(F*FY)/(C*FX)
680:X=MX*FX:Y=M*X: IF Y>YW THEN LET Y=Y
    W:X=Y/M
690:LPRINT CHR$ 27;"1": LPRINT "D";X;",
    ";Y
700:Y=D*FY:X=MX*FX:S=Y
710:IF Q=Y THEN 780
720:Z$= STR$ (D):R=12: IF LEN (Z$)>=6
    THEN LPRINT CHR$ 27;"?a":R=6
730:IF L=0 OR Y>Q+R OR Y<Q-R THEN LET Q
    =Y: GOTO 760
740:IF Y>Q THEN LET Q=Q+R: GOTO 760
750:IF Y<Q THEN LET Q=Q-R
760:LPRINT CHR$ 27;"2": LPRINT "M";-
    LEN (Z$)*R;",";Q-R/2: LPRINT "P";Z$
770:LPRINT CHR$ 27;"?b"
780:LPRINT CHR$ 27;"2": LPRINT "M";0;",
    ";Y: LPRINT "J";X;",";Q
790:X=MX*FX:Y=N*X+S: IF Y>YW THEN LET Y
    =YW:X=(Y-S)/N
800:LPRINT "M";0;",";S: LPRINT "D";X;",
    ";Y
```

```
810:X=H*FX:Y=G*FY: IF X>XW OR Y>YW THEN
    950
820:IF P=Y THEN 890
830:Z$= STR$ (G):R=12: IF LEN (Z$)>=o
    THEN LPRINT CHR$ 27;"?a":R=6
840:IF L=0 OR Y>P+R OR Y<P-R THEN LET P
    =Y: GOTO 870
850:IF Y>P THEN LET P=P+R: GOTO 870
860:IF Y<P THEN LET P=P-R
870:LPRINT CHR$ 27;"3": LPRINT "M";-
    LEN (Z$)*R;",";P-R/2: LPRINT "P";Z$
880:LPRINT CHR$ 27;"?b"
890:LPRINT CHR$ 27;"3": LPRINT "M";0;",
    ";Y: LPRINT "D";X;",";Y;",";X;",";0
900:IF O=X THEN 950
910:Z$= STR$ (H)
920:IF L=0 THEN LPRINT "M";X-( LEN (Z$)
    *6);",";-14: GOTO 940
930:LPRINT "M';X-( LEN (Z$)*6);",";-28
940:LPRINT "P";Z$:O=X
950:Y=-53:X=105: IF L=3 THEN LET X=230
960:LPRINT CHR$ 27;"0"
970:S=A: GOSUB 1120
980:S=B: GOSUB 1120
990:S=C: GOSUB 1120
1000:S=D: GOSUB 1120
1010:S=E: GOSUB 1120
1020:S=F: GOSUB 1120
1030:S=G: GOSUB 1120
1040:S=H: GOSUB 1120
1050:S=K: GOSUB 1120
1060:IF L=3 THEN 1160
1070:LPRINT "H"
1080:L=3: CLS : INPUT "Simulation ?(Y/
     N)=>";Z$
1090:IF Z$="Y" OR Z$="y" THEN 140
1100:IF Z$="N" OR Z$="n" THEN 1160
1110:GOTO 1080
1120:REM  * Table Print
1130:Y=Y-10:T=(10- LEN ( STR$ S))*12+2
1140:LPRINT "M";X+T;",";Y: LPRINT "P";
     STR$ S
1150:RETURN
1160:CLS : LPRINT "M";-80;",";-300:
     LPRINT CHR$ 27;"a"
1170:END
```

2813 bytes

■ MEMORY CONTENTS

A	sales unit price	S	√
B	variable unit price	T	for work
C	sales quantity	X	X-coordinates
D	fixed cost	Y	Y-coordinates
E	sales amount	Z$	√
F	variable cost	XW	width of graph
G	break-evenpoint (amount)	YW	height of graph
H	break-evenpoint (quantity)	MX	maximum value on X-axis
I	loop counter	MY	maximum value on Y-axis
J	√	FX	scale factor of X-axis
K	profit	FY	scale factor of Y-axis
L	type of lines	SX	X-coordinates (left side of the table)
M	slant of sales line	EX	X-coordinates (right side of the table)
N	slant of cost line	SY	Y-coordinates (upper side of the table)
O	√	EY	Y-coordinates (lower side of the table)
P	√	SW	flag
Q	√	M$(Ø)	item description
R	√		

Program Title: **ANALYSIS GRAPH FOR CALCULATION OF PROFIT AND LOSS**

The graph clearly shows you the status of profit and loss of sales. Some of the enterprises prepare Profit and Loss Statements to compare the balance between incomes and expenditures.

However this statement is not so comprehensive because it is only a collection of many figures.

This program is in the form of a graph displaying the corelationship of sales amounts, expenditures and profits. You can immediately grasp the status of profit and loss in your business, and this will serve as some the most useful data for establishing a policy for managing your business.

▪ HOW TO OPERATE

1. Program starts with R U N ENTER .
   First, input the company name (within compiled limit of 15 letters) then input the date of the data in 8 figures as "YY MM DD".
   Subsequently, each item (9 items) is displayed on the screen. Input the amount or the value of the percentage. The number displayed before the description of the item is the number for each item.
2. When the input is complete, decide whether data is required or not. In the case of modification, operate Y ENTER and input the item number which you want to modify. When the description of the item is displayed, input the data.
   Operate N ENTER when no data modification is required. Then Analysis Graph for Calculation of Profit and Loss will be displayed. Program ends.

   Note: Net Sales Amount < Cost of Sales + Total Profit
   Net Sales Amount < Cost of Materials + Labor Cost +
   Expenses + Selling Cost + Operating Profit

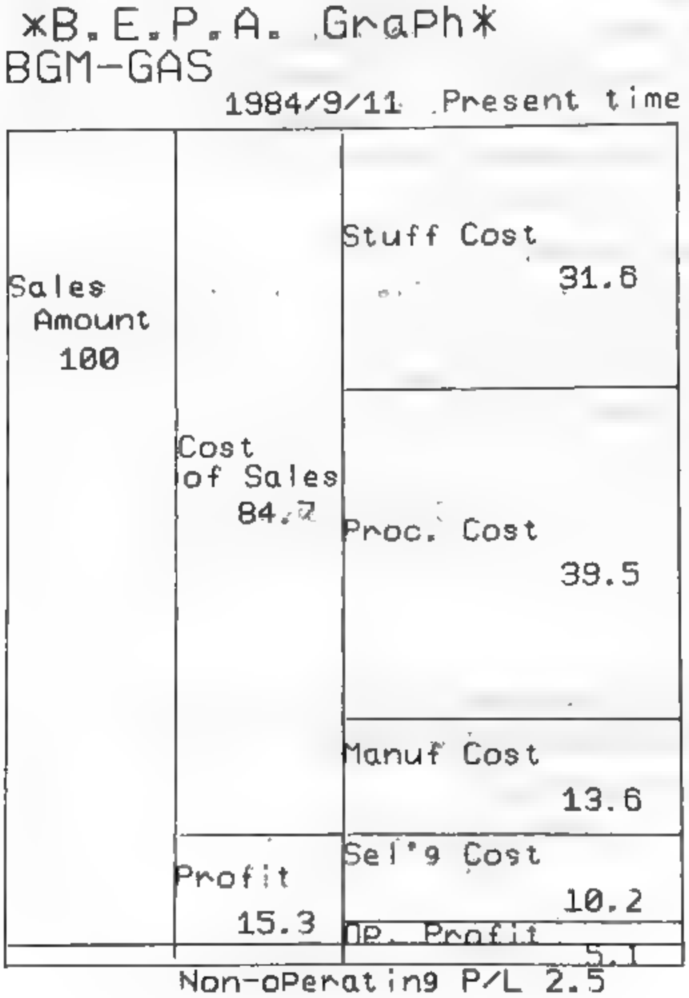
▪ EXAMPLE

With the following data, write out the Analysis Graph of Profit and Loss Calculation. The unit of the amount is based on 5 digits (10000).

Company Name	:	BGM-GAS
Date	:	1984, 9 (September), 11th

No.	Item	Amount	Percentage
1	Sales Amount	4500.0	100
2	Cost of Sales	3811.5	84.7
3	Profit	688.5	15.3
4	Stuff Cost	1422.0	31.6
5	Process Cost	1777.5	39.5
6	Manuf Cost	612.0	13.6
7	Sel'g Cost	459.0	10.2
8	Operating Profit	229.5	5.1
9	Non-Operating Profit and Loss	112.5	2.5

■ **PRINTOUT (Colored Refer to page 3)**



```
  *B.E.P.A. Graph*
BGM-GAS
              1984/9/11 Present time
```

■ KEY OPERATION SEQUENCE

1. R U N ENTER
[Program starts]

```
  ** B.E.P.A. Graph **
Company Name
          =>_
```

2. B G M – G A S
ENTER [Company name input]

```
  ** B.E.P.A. Graph **
Company Name
          => BGM-GAS
Date      =>?   Y   M   D
```

3. 1984 ENTER 9 ENTER 11 ENTER
[Graph compiled date input]

```
1. Sales Amount
 =>_
```

4. 4500 ENTER [Sales amount input]

```
1. Sales Amount
 =>4500
2. Cost of Sales
 =>_
```

Input same as above

5. 112.6 ENTER [Non-operating
profit and loss input]

```
 =>229.5
9. Non-operating P/L
 =>112.6
Correction? (Y/N) =>_
```

6. Y ENTER [Data modification]

```
9. Non-operating P/L
 =>112.6
Correction? (Y/N) =>Y
Number =>_
```

7. 9 ENTER [Non-operating profit
and loss and modification of data]

```
Correction? (Y/N) =>
Number =>9
9. Non-operating P/L
 =>_
```

8. 112.5 ENTER [Correct data input]

```
Number =>9
9. Non-operating P/L
 =>
Correction? (Y/N) =>_
```

9. N ENTER

```

      ** Printout **
```

```

>
```

(Analysis Graph of Profit and Loss
Calculation is displayed. Program
ends.)

■ PROGRAM LIST

```
 10:CLEAR : CLS : CONSOLE 39: LPRINT :
    WAIT 0: LPRINT CHR$ 27;"a"; CHR$ 13
 20:DIM DT(8,1),KN$(1)*40,KM$(8)*26
 30:REM * Initialize
 40:FOR I=0 TO 8: READ KM$(I): NEXT I
 50.DATA "Sales Amount"."Cost of Sales"
    ."Profit"
 60:DATA "Stuff Cost". Proc. Cost"."Man
    uf Cost","Sel'g Cost"
 70:DATA "Op. Profit","Non-operating P/
    L"
 80:PRINT "** B.E.P.A. Graph **"
 90:PRINT "Company Name": INPUT " =>";K
    N$(0)
100:IF LEN KN$(0)>15 CLS - GOTO 80
110.PRINT "Date  =>     Y    M    D"
120.CURSOR 8,3. INPUT Q$
130.T= VAL Q$
140.CURSOR 15,3. INPUT R$.J= VAL R$
150.IF U<1 OR U>12 THEN 140
160.CURSOR 20,3. INPUT S$.U= VAL S$
170.IF V<1 OR V>31 THEN 160
180.KN$(1)=Q$+"/"+R$+"/"+S$+"  .  CLS
190.FOR I=0 TO 8. PRINT I+1,KM$(I).
    INPUT " =>".DT(I,0). NEXT I
200.INPUT "Connection? (Y/N) =>",A$
210.IF A$="Y" OR A$= y" THEN 240
220.IF A$="N" OR A$= n" THEN 280
230.GOTO 200
240.INPUT "Number =>".J
250.IF J>9 OR J<1 THEN 240
260.PRINT J,KM$(J 1). INPUT "=>".DT(J-1
    ,0)
270.GOTO 200
280.CLS . CURSOR 5,1. PRINT "** Printou
    t **"
290.IF DT(0,0)<DT(1,0)+DT(2,0) THEN 320
300.IF DT(0,0)<DT(3,0)+DT(4,0)+DT(5,0)+
    DT(6,0)+DT(7,0) THEN 320
310.GOTO 330
320.CLS . WAIT 100. CURSOR 0,1. PRINT "
    * Data is inconsistent *": WAIT 0:
    CLS - GOTO 190
330:FOR I=0 TO 8.DT(I,1)=( INT (DT(I,0)
    /DT(0,0)*1000+.5))/10. NEXT I
340:LPRINT CHR$ 27; 0": LPRINT CHR$ 27;
    "b": LPRINT "L0"
350.LPRINT CHR$ 27; ?o": LPRINT "M";60;
    ",";-20
360:LPRINT "P";"*B.E.P.A. Graph*"
370:LPRINT "M";50;",";-45: LPRINT "P";K
    N$(0): LPRINT CHR$ 27;"?b"
380:LPRINT "M";180;",";-65: LPRINT "P";
    KN$(1);" Present time"
390.LPRINT "M";450;",";-75
400:LPRINT "J";0;",";-500;",";-400;".";
    0;",";0;",";500;",";400;",";0
410:LPRINT "R";-200;",";0: LPRINT "J";0
    ;",";-500
420:LPRINT "R";-100;",";0: LPRINT "J";0
    ;",";500
430:LPRINT "M";50;",";-575: LPRINT "I":
    LPRINT CHR$ 27;"1"
440:X=2:Y=400
450:LPRINT "M";X;",";Y
460:LPRINT "P"; MID$ (KM$(0),1,6)
470:LPRINT "M";X+15;",";Y-20
480:LPRINT "P"; MID$ (KM$(0),7,6)
490:Y=Y-44: LPRINT "M";32;",";Y: LPRINT
    "P"; STR$ (DT(0,1))
500:J=DT(1,1)*5:X=103:Y=500-(J-52)/2-12
510:LPRINT "M";X;",";Y: LPRINT "P";
    MID$ (KM$(1),1,4)
520:X=106:Y=Y-18
530:LPRINT "M";X;",";Y: LPRINT "P";
    MID$ (KM$(1),6,8)
540:X=138:Y=Y-22
550:LPRINT  M";X; ,";Y: LPRINT "P";
    STR$ (DT(1,1))
560:SY=DT(2,1)*5: LPRINT "M";100;",";SY
    : LPRINT "J";100;",";0
570:A=27:H=40: IF DT(2,1)<8 THEN LET A=
    12:H=24
580:W=SY:Y=SY-(W-H)/2-12:J=SY
590:LPRINT "M";102;",";Y: LPRINT "P";KM
    $(2)
600:Y=Y-A:X=200-62: IF DT(2,1)<10 THEN
    LET X=X+12
610:LPRINT "M";X;",";Y: LPRINT "P";
    STR$ (DT(2,1))
620:SY=500: J=0
630:FOR I=3 TO 7:C=DT(I,1)
640:IF C=0 THEN 790
650:W=C*5:EY=SY-W
660:IF I=7 THEN 680
670:LPRINT "M";200;",";EY: LPRINT "J";2
    00;",";0
680:A=27:H=40: IF C<8 THEN LET A=12:H=2
    4: IF C<4.8 THEN LET A=0:H=12
690:IF W<12 THEN 740
700:Y=SY-(W-H)/2-12:X=202: LPRINT "M";X
    ;",";Y: LPRINT "P";KM$(I)
710:Y=Y-A:X=400-68: IF C<10 THEN LET X=
    X+12
720:LPRINT "M";X;",";Y: LPRINT "P";
    STR$ C
730:SY=EY: GOTO 790
740:J=J+1:B$="A": IF J=2 THEN LET B$="B
    "
750:X=403:Y=SY-(W-H)/2-12: LPRINT "M";X
    ;",";Y: LPRINT "P";B$
```

```
760:X=20:Y=-40-(J-1)*15
770:LPRINT "M";X;",";Y: LPRINT "P";B$;"
    ";KM$(I);"  ": STR$ C
780:GOTO 730
790:NEXT I
800:IF DT(8,1)=0 THEN 850
810:LPRINT CHR$ 27;"2"
820:Y=DT(8,1)*5: LPRINT "M";0;",";Y:
    LPRINT "J";400;",";0
830:LPRINT CHR$ 27;"1"
840:LPRINT "M";105;",";-15: LPRINT "P";
    KM$(8);" ": STR$ (DT(8,1))
850:LPRINT "M";-50;",";-200
860:LPRINT CHR$ 27;"a": CLS
870:END
```

2389 bytes

■ MEMORY CONTENTS

A, A$	for work, for input
B$	for work
C	√
H	item description, width of plot
I	loop counter
J	√
Q$	year
R$	month
S$	day
T	year
U	month
V	day
W	item description, width of plot
X	X-coordinates
Y	Y-coordinates
SY	division of the graph (Y-coordinates)
EY	division of the graph (Y-coordinates)
DT(8,1)	input amount and percentage
KN$(1)	company name, compiled data for input
KM$(8)	item description

Program Title: MATRIX OPERATION

Computers help you to operate a difficult matrix!

A matrix series is indispensable for technical calculations and the disposal of statistics. However, these Matrix are quite hard to operate. The computer then assists you to execute three operations of addition, subtraction and multiplication, thereby displaying every result in the form of Matrix on the screen.

■ HOW TO OPERATE

1. R U N ENTER (Program starts.)
   When three kinds of operations (addition, subtraction and multiplication) are displayed, input the operation number which you want to execute. When the program is completed, operate 4 ENTER .
2. Input the row and column numbers of the Matrix.
   - In the case of addition and subtraction, input the row and column numbers only once, since the row and column numbers of Matrix A and B are equivalent.
   - In the case of multiplication, input the column number only for Matrix B since the row number for Matrix B is decided at the time when the row and column numbers for Matrix A are input. The row number for Matrix A equals the column number for Matrix B.
3. Input the data of each Matrix.
   In accordance with the display, input the data of the Matrix. When the data is input, the results are output to the printer.
   When the output is completed, the program will return to the initial screen of the menu.
   - When data is input incorrectly, operate ENTER and the input of the data prior to the last data stands by. Then the data can be modified.

Note: Maximum input capacity of the row and column numbers is as follows.

- Standard capacity . . . . up to 3 rows · 3 columns Max.
- When attached with CE-201M. . . . . up to 17 rows · 20 columns Max.
- When attached with CE-202M. . . . . up to 17 rows · 40 columns Max.

(However, the width of the printing paper is limited to 17 rows.)

■ CONTENTS

Matrix Basic operation method

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \qquad B = \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

$$A + B = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

$$A - B = \begin{bmatrix} a_{11} - b_{11} & a_{12} - b_{12} \\ a_{21} - b_{21} & a_{22} - b_{22} \end{bmatrix}$$

$$A * B = \begin{bmatrix} a_{11} \cdot b_{11} + a_{12} \cdot b_{21} & a_{11} \cdot b_{12} + a_{12} \cdot b_{22} \\ a_{21} \cdot b_{11} + a_{22} \cdot b_{21} & a_{21} \cdot b_{12} + a_{22} \cdot b_{22} \end{bmatrix}$$

■ EXAMPLE

Execute the following two operations.

1. Addition

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix}$$

2. Multiplication

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \times \begin{bmatrix} 9 & 8 & 7 \\ 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix}$$

■ PRINTOUT

In case of example 1

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} + \begin{bmatrix} 9 & 8 & 7 \\ 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix} = \begin{bmatrix} 10 & 10 & 10 \\ 5 & 7 & 9 \\ 13 & 13 & 13 \end{bmatrix}$$

In case of example 2

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} * \begin{bmatrix} 9 & 8 & 7 \\ 1 & 2 & 3 \\ 6 & 5 & 4 \end{bmatrix} = \begin{bmatrix} 29 & 27 & 25 \\ 77 & 72 & 67 \\ 125 & 117 & 109 \end{bmatrix}$$

■ KEY OPERATION SEQUENCE

《Example 1》

1. R U N ENTER
[Program starts]

```
   ** Matrix Operation **
     1: A + B    2: A − B
     3: A *B     4: E N D
     Selection =>_
```

2. 1 ENTER [Select addition in operation]

```
Row number of A, B = _
```

3. 3 ENTER [Input of row number of Matrix A, B]

```
Row number of A, B = 3
Column number of A, B = _
```

4. 3 ENTER [Input of column number of Matrix A, B]

```
Input data for A
(1, 1)  =>_
```

5. 1 ENTER [Input of data for a_{11} of Matrix A]

```
Input data for A
(1, 1)    =>1
(1, 2)    =>_
```

6. 3 ENTER [Input of other data]

```
Input data for A
(1, 1)    =>1
(1, 2)    =>3
(1, 3)    =>_
```

7. ENTER [Data modification]

```
(1, 2)    =>_
```

8. 2 ENTER [Correct data input]

```
(1, 2)    =>2
(1, 3)    =>_
```

9. 3 ENTER [Input a_{13} data of Matrix A]

```
(1, 2)    =>2
(1, 3)    =>3
(2, 1)    =>_
```

⋮

Input the following data of Matrix A and B as above.

⋮

10. 4 ENTER [Input b_{33} data of Matrix B]

```
   ** Matrix Operation **

         ** Printout **
```

(Results are printed out)

```
   ** Matrix Operation **
     1: A + B    2: A − B
     3: A *B     4:  E N D
     Selection =>_
```

11. 4 ENTER [Program ends]

```
>
```

■ KEY OPERATION SEQUENCE

《Example 2》

1. [R] [U] [N] [ENTER]
   [Program starts]

```
     ** Matrix Operation **
       1: A + B    2: A − B
       3: A * B    4: E N D
       Selection =>_
```

2. 3 [ENTER] [Select multiplication in operation]

```
Row number of A = _
```

3. 3 [ENTER] [Input of row number of Matrix A]

```
Row number of A = 3
Column number of A = _
```

4. 3 [ENTER] [Input of column number of Matrix A]

```
Row number of A = 3
Column number of A = 3
Row number of B = 3
Column number of B = _
```

5. 3 [ENTER] [Input of column number of Matrix B]

```
Input data for A
(1, 1)    =>_
```

6. 1 [ENTER] [Input of data for a_{11} of Matrix A]

```
Input data for A
(1, 1)    =>1
(1, 2)    =>_
```

7. 3 [ENTER] [Input of other data]

```
Input data for A
(1, 1)    =>1
(1, 2)    =>3
(1, 3)    =>_
```

8. [ENTER] [Data modification]

```
(1, 2)    =>_
```

9. 2 [ENTER] [Correct data input]

```
(1, 2)    =>2
(1, 3)    =>_
```

10. 3 [ENTER] [Input a_{13} data of Matrix A]

```
(1, 2)    =>2
(1, 3)    =>3
(2, 1)    =>_
```

⋮

Input the following data of Matrix A and B as above.

⋮

11. 4 [ENTER] [Input b_{33} data of Matrix B]

```
     ** Matrix Operation **

          ** Printout **
```

(Results are printed out)

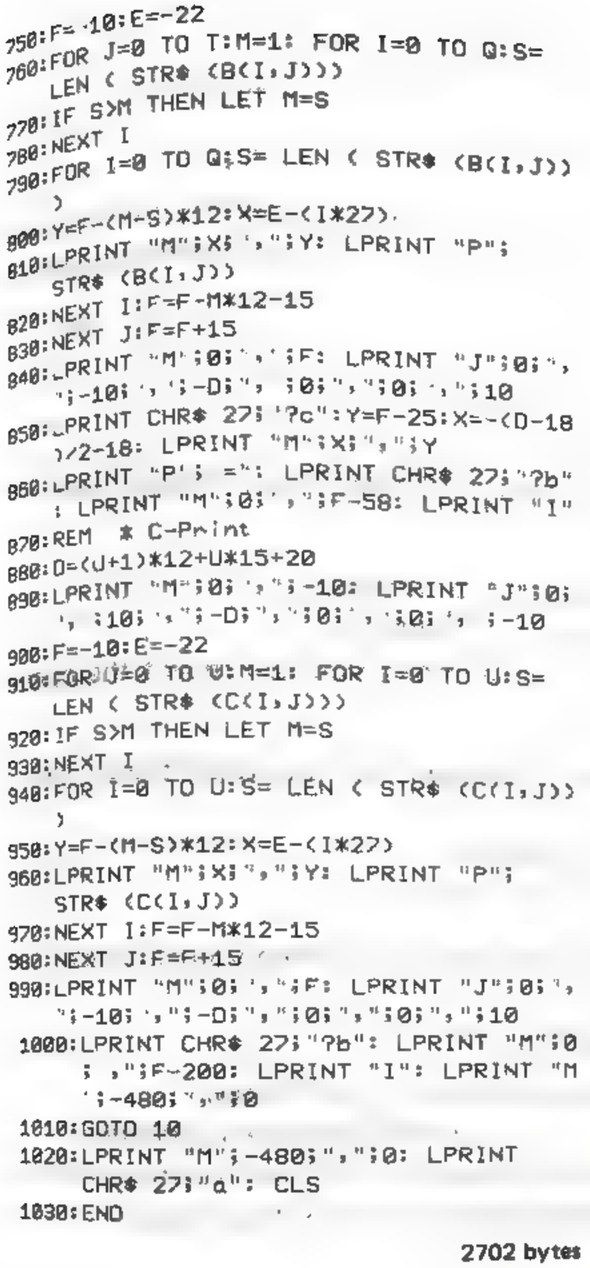
```
     ** Matrix Operation **
       1: A + B    2: A − B
       3: A * B    4: E N D
       Selection =>_
```

12. 4 [ENTER] [Program ends]

```

>
```

■ PROGRAM LIST

```
10:LPRINT CHR$ 27;"0": LPRINT CHR$ 27;
   "a": LPRINT CHR$ 13: LPRINT CHR$ 27
   ;"b": LPRINT "L0"
20:LPRINT "Q1"
30:CLEAR : CLS : WAIT 0: CONSOLE 39:
   LPRINT
40:CURSOR 0,0: PRINT "** Matrix Operat
   ion **"
50:CURSOR 2,1: PRINT "1:A + B  2:A - B
   "
60:CURSOR 2,2: PRINT "3:A * B  4:E N D
   "
70:CURSOR 2,3: INPUT "Selection =>";N:
   CLS
80:IF N=4 THEN 1020
90:IF N>4 OR N<1 THEN 30
100:IF N=3 THEN 140
110:INPUT "Row number of A,B=";O: IF O<
    1 THEN 110
120:INPUT "Column number of A,B=";P: IF
    P<1 THEN 120
130:O=O-1:P=P-1:Q=O:T=P:U=O:V=P: GOTO 1
    90
140:INPUT "Row number of A=";O: IF O<1
    THEN 140
150:INPUT "Column number of A=";P: IF P
    <1 THEN 150
160:Q=P: PRINT "Row number of B="; STR$
    Q
170:INPUT "Column number of B=";T: IF T
    <1 THEN 170
180:O=O-1:P=P-1:Q=Q-1:T=T-1:U=O:V=T
190:DIM A(O,P),B(Q,T),C(U,V)
200:CLS : PRINT "Input data for A":L=0
210:FOR I=0 TO O: FOR J=0 TO P
220:L=L+1: IF L>3 THEN CLS :L=0
230:CURSOR 0,L: PRINT "("; STR$ (I+1);"
    ,"; STR$ (J+1);")"
240:CURSOR 7,L:A$="": INPUT "=>";A$
250:IF LEN A$>0 THEN LET A(I,J)= VAL A$
    :G=I:H=J: GOTO 290
260:L=L+1: IF L>3 THEN CLS :L=0
270:CURSOR 0,L: PRINT "("; STR$ (G+1);"
    ,"; STR$ (H+1);")"
280:CURSOR 7,L: INPUT "=>";A(G,H): GOTO
    220
290:NEXT J: NEXT I
300:CLS : PRINT "Input data for B":L=0
310:FOR I=0 TO Q: FOR J=0 TO T
320:L=L+1: IF L>3 THEN CLS :L=0
330:CURSOR 0,L: PRINT "("; STR$ (I+1);"
    ,"; STR$ (J+1);")"
340:CURSOR 7,L:A$="": INPUT "=>";A$
350:IF LEN A$>0 THEN LET B(I,J)= VAL A$
    :G=I:H=J: GOTO 390
360:L=L+1: IF L>3 THEN CLS :L=0
370:CURSOR 0,L: PRINT "("; STR$ (G+1);"
    ,"; STR$ (H+1);")"
380:CURSOR 7,L: INPUT "=>";B(G,H): GOTO
    320
390:NEXT J: NEXT I
400:LPRINT "M";480;",";0: LPRINT "I":
    LPRINT CHR$ 27;"?b"
410:CLS : CURSOR 1,0: PRINT "** Matrix
    Operation **": CURSOR 2,2: PRINT "*
    * Printout **"
420:ON N GOSUB 440,470,500
430:GOTO 540
440:REM  * Addition
450:FOR I=0 TO O: FOR J=0 TO P:C(I,J)=A
    (I,J)+B(I,J)
460:NEXT J: NEXT I: RETURN
470:REM  * Subtraction
480:FOR I=0 TO O: FOR J=0 TO P:C(I,J)=A
    (I,J)-B(I,J)
490:NEXT J: NEXT I: RETURN
500:REM  * Multiplication
510:FOR I=0 TO O: FOR J=0 TO T: FOR K=0
    TO Q
520:C(I,J)=C(I,J)+A(I,K)*B(K,J)
530:NEXT K: NEXT J: NEXT I: RETURN
540:REM  * A-Print
550:D=(O+1)*12+O*15+20
560:LPRINT "M";0;",";-10: LPRINT "J";0;
    ",";10;",";-D;",";0;",";0;",";-10
570:F=-10:E=-22
580:FOR J=0 TO P:M=1: FOR I=0 TO O:S=
    LEN ( STR$ (A(I,J)))
590:IF S>M THEN LET M=S
600:NEXT I
610:FOR I=0 TO O:S= LEN ( STR$ (A(I,J))
    )
620:Y=F-(M-S)*12:X=E-(I*27)
630:LPRINT "M";X;",";Y: LPRINT "P";
    STR$ (A(I,J))
640:NEXT I:F=F-M*12-15
650:NEXT J:F=F+15
660:LPRINT "M";0;",";F: LPRINT "J";0;",
    ";-10;",";-D;",";0;",";0;",";10
670:LPRINT CHR$ 27;"?c":Y=F-25:X=-(D-18
    )/2-10: LPRINT "M";X;",";Y
680:IF N=1 THEN LPRINT "P";"+"
690:IF N=2 THEN LPRINT "P";"-"
700:IF N=3 THEN LPRINT "P";"*"
710:LPRINT CHR$ 27;"?b": LPRINT "M";0;"
    ,";F-58: LPRINT "I"
720:REM  * B-Print
730:D=(Q+1)*12+Q*15+20
740:LPRINT "M";0;",";-10: LPRINT "J";0;
    ",";10;",";-D;",";0;",";0;",";-10
```



```
750:F=-10:E=-22
760:FOR J=0 TO T:M=1: FOR I=0 TO Q:S=
    LEN ( STR$ (B(I,J)))
770:IF S>M THEN LET M=S
780:NEXT I
790:FOR I=0 TO Q:S= LEN ( STR$ (B(I,J))
    )
800:Y=F-(M-S)*12:X=E-(I*27)
810:LPRINT "M";X;",";Y: LPRINT "P";
    STR$ (B(I,J))
820:NEXT I:F=F-M*12-15
830:NEXT J:F=F+15
840:LPRINT "M";0;",";F: LPRINT "J";0;",
    ";-10;",";-D;",";0;",";0;",";10
850:LPRINT CHR$ 27;"?c":Y=F-25:X=-(D-18
    )/2-18: LPRINT "M";X;",";Y
860:LPRINT "P";"=": LPRINT CHR$ 27;"?b"
    : LPRINT "M";0;",";F-58: LPRINT "I"
870:REM  * C-Print
880:D=(J+1)*12+U*15+20
890:LPRINT "M";0;",";-10: LPRINT "J";0;
    ",";10;",";-D;",";0;",";0;",";-10
900:F=-10:E=-22
910:FOR J=0 TO V:M=1: FOR I=0 TO U:S=
    LEN ( STR$ (C(I,J)))
920:IF S>M THEN LET M=S
930:NEXT I
940:FOR I=0 TO U:S= LEN ( STR$ (C(I,J))
    )
950:Y=F-(M-S)*12:X=E-(I*27)
960:LPRINT "M";X;",";Y: LPRINT "P";
    STR$ (C(I,J))
970:NEXT I:F=F-M*12-15
980:NEXT J:F=F+15
990:LPRINT "M";0;",";F: LPRINT "J";0;",
    ";-10;",";-D;",";0;",";0;",";10
 1000:LPRINT CHR$ 27;"?b": LPRINT "M";0
      ;",";F-200: LPRINT "I": LPRINT "M
      ";-480;",";0
 1010:GOTO 10
 1020:LPRINT "M";-480;",";0: LPRINT
      CHR$ 27;"a": CLS
 1030:END
```

2702 bytes

■ MEMORY CONTENTS

A$	key-in
D	width of x-coodinates when printing the matrix
E	X-coordinates (start of printout)
F	Y-coordinates (start of printout)
G	row number
H	column number
I	loop counter
J	loop counter
K	loop counter
L	counter of displayed row number
M	maximum letter's number of each column
N	types of operation
O	row number of matrix A
P	column number of Matrix A
Q	row number of Matrix B
S	letter number of each column
T	column number of Matrix B
U	row number of Matrix C
V	column number of Matrix C
X	numerical figure's plot (X-coordinates)
Y	numerical figure's plot (Y-coordinates)
A(O,P)	Matrix A
B(Q,T)	Matrix B
C(U,V)	results of operation, Matrix C

Program Title:	N-DEGREE EQUATION

The computer helps you solve equations easily!

An equation which has been difficult to solve can be easily solved by this program employing the NEWTON method. Simply input the maximum degree number of $f(x)$ and the coefficient of each degree number.

■ HOW TO OPERATE

1. Program starts with R U N ENTER .
   Input the degree number and its coefficient in order in accordance with the display.
2. When the input of the data is completed, $f(x)$ is displayed.
   - If the displayed data is correct, operate Y ENTER .
   - If the data is incorrect, operate N ENTER then you can repeat the operation from the first step again.
3. Select whether you want to display the data you have input on the screen or output the results to the printer.
   - Operate Y ENTER for output to the printer.
   - Operate N ENTER for the output to the display.

   Then the root is over 3 different ways, press ENTER and then you can check the result in and after 4.
   When the output is completed, this program ends.

   Note: 1. The maximum capacity of degree number is as follows;
   - Standard capacity . . . . up to 34 degree numbers

   2. This program adopts the basic algorithm of the Newton method. Multipled root may be obtained, but it occurs that one part of the root is not displayed.

■ REFERENCE

1. $X_{n+1} \doteq X_n - \dfrac{f(X_n)}{f'(X_n)}$

   When the absolute value of the differential between X_n and X_{n+1} is below 10^{-8}, round off the fraction of the value to 10^{-9}. The $f'(x)$ of the 1st differentiation is defined here as follows.

   $$f'(x) = n \cdot a_n x^{n-1} + (n-1)\, a_{n-1} x^{n-2} + \cdots\cdots 2a_2 x + a_1$$

2. The above starting point is automatically change to $X\emptyset = (-2)^{i} = 1, -2, 4, -8, 16$ ($i = \emptyset, 1, 2, 3, 4$). When you want these changing proceeds to change, modify –2 of 260 lines accordingly.

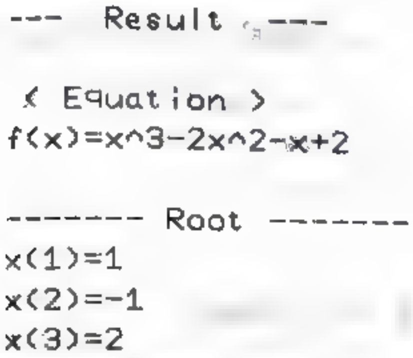
3. When changing 10^{-8}, modify E = .ØØØ1 of 30 line.

■ EXAMPLE

Question: $x^3 - 2x^2 - x + 2 = 0$

Root: (–1, 1, 2)

■ PRINTOUT



```
---  Result  ---

 < Equation >
f(x)=x^3-2x^2-x+2

------- Root -------
x(1)=1
x(2)=-1
x(3)=2
```

■ KEY OPERATION SEQUENCE

1. R U N ENTER
[Program starts]

```
   ** n-Degree Equation **
Degree = _
```

2. 3 ENTER [Degree No. input]

```
   ** n-Degree Equation **
Degree = 3
a (3) = ?
```

3. 2 ENTER [Coefficient a_3 input]

```
   ** n-Degree Equation **
Degree = 3
a (3) = 2
a (2) = ?
```

:

Input the coefficients as above.

:

4. 2 ENTER [Coefficient a_0 input]

```
f (x) = 2x ^ 3 – 2x ^ 2 – x + 2
(Y/N)
OK?
?
```

5. N ENTER [Modification as data is not correct]

```
   ** n-Degree Equation **
Degree = _
```

6. 3 ENTER

```
   ** n-Degree Equation **
Degree = 3
a (3) = ?
```

7. 1 ENTER [Correct data input]

```
   ** n-Degree Equation **
Degree = 3
a (3) = 1
a (2) = ?
```

:

Input the following coefficient same as above.

:

8. 2 ENTER [Coefficient a_0 input]

```
f (x) = x ^ 3 – 2x ^ 2 – x + 2
(Y/N)
OK?
?
```

9. Y ENTER

```
   ** n-Degree Equation **

Print the result (Y/N)?
```

10. Y ENTER [Output the result to the printer]

```

      ** Under Process **
```

```

>
```

(When the result is output, the program ends.)

▪ PROGRAM LIST

```
10:CONSOLE 39: LPRINT CHR$ 27;"?b"
20:CLS : CLEAR : WAIT 0: PRINT = PRINT
30:E=.0001:E1=E/10000
40:REM  * Data input *
50:CURSOR 0,0: PRINT "** n-Degree Equa
   tion **"
60:CURSOR 0,1: INPUT "Degree =";N
70:DIM W(N),X(N),C$(N+1)*10,F$(0)*10,S
   $(0)*10:L=1
80:FOR I=N TO 0 STEP -1
90:L=L+1: IF L>3 LET L=0: CLS
100:CURSOR 0,L: PRINT "a("; STR$ I;")="
    CURSOR 7,L: INPUT W(I)
110:IF W(N)=0 LET I=0: NEXT I: GOSUB 11
    10: GOTO 20
120:NEXT I
130:CLS : GOSUB 580
140:CURSOR 0,1: PRINT "(Y/N)": CURSOR 1
    ,2: PRINT "OK?": CURSOR 2,3: INPUT
    Y$
150:IF Y$="Y" OR Y$="y" THEN 180
160:IF Y$="N" OR Y$="n" THEN 20
170:GOTO 140
180:CLS : CURSOR 0,0: PRINT "** n-Degre
    e Equation **": CURSOR 0,2
190:PRINT "Print the result (Y/N) ":
    CURSOR 22,2: INPUT Y$
200:IF Y$="Y" OR Y$="y" OR Y$="N" OR Y$
    ="n" GOTO 220
210:GOTO 180
220:REM  * Calculation *
230:CLS : CURSOR 2,1: PRINT "** Under P
    rocess **"
240:K=0: FOR I=0 TO 4
250:IF K=N LET I=5: GOTO 430
260:X0=(-2)^I:X=X0
270:IF X=0 GOSUB 850: GOTO 290
280:GOSUB 800
290:IF F1=0 THEN 370
300:IF X=0 GOSUB 930: GOTO 330
310:GOSUB 880
320:IF F2=0 THEN 430
330:X=X-F1/F2
340:IF X=0 THEN 360
350:IF ABS (F1/F2/X)>=1 THEN PRINT F1,F
    2,X, ABS (F1/F2/X): GOTO 430
360:IF ABS (F1/F2)>=E1 THEN 270
370:X= INT (X/E+.5)*E
380:IF K=0 THEN 420
390:FOR J=1 TO K
400:IF X=X(J) LET J=K: NEXT J: GOTO 430
410:NEXT J
420:K=K+1:X(K)=X
430:NEXT I
440:REM  * Output *
450:LPRINT CHR$ 27;"0": LPRINT CHR$ 27;
    "a"
460:IF Y$="Y" OR Y$="y" THEN PRINT =
    LPRINT
470:WAIT 50: BEEP 3: CLS : CURSOR 4,1:
    PRINT "---  Result  ---": PRINT ""
480:IF Y$="N" OR Y$="n" THEN 510
490:WAIT 0: CLS : PRINT " < Equation >
    "
500:GOSUB 960: LPRINT : LPRINT
510:WAIT 0: CLS : PRINT "-------- Root -
    -------"
520:IF K=0 THEN WAIT : PRINT "       Not
     Real !": GOTO 570
530:CURSOR 24: FOR I=1 TO K: WAIT 0:I1=
    INT (I/4): IF I-4*I1=3 THEN WAIT
540:IF I-4*I1=0 THEN CLS
550:PRINT "x("; STR$ I;")="; STR$ X(I)
560:NEXT I: WAIT : PRINT
570:CLS : END
580:REM  * f(x) Print *
590:PRINT "f(x)=":L=5:Q=0:O=5
600:FOR I=N TO 0 STEP -1:C$(0)="":B$=""
610:IF W(I)=0 THEN 780
620:IF ABS W(I)=1 AND I<>0 LET C$(0)=""
    : GOTO 640
630:C$(0)= STR$ ABS W(I)
640:IF I=N THEN 660
650:IF W(I)>0 LET B$="+"
660:IF W(I)<0 LET B$="-"
670:C$(0)=B$+C$(0)
680:IF I=0 THEN 710
690:IF I=1 LET C$(0)=C$(0)+"x": GOTO 71
    0
700:C$(0)=C$(0)+"x^"+ STR$ I
710:O=O+ LEN C$(0)
720:IF O>23 AND Q=3 GOSUB 1050:O=O+ LEN
    C$(0): WAIT 0: GOTO 740
730:IF O>23 GOSUB 1090: GOTO 750
740:CURSOR (O- LEN C$(0)),Q: PRINT C$(0
    )
750:M=M+1
760:C$(M)=C$(0)
770:S$(0)=C$(0)
780:NEXT I
790:RETURN
800:REM  * f(x) *
810:F1=0: FOR J=0 TO N
820:F1=F1+W(J)*X^J
830:NEXT J
840:RETURN
850:REM  * f(0) *
860:F1=W(0)
```

```
870:RETURN
880:REM  * f'(x) *
890:F2=0: FOR J=1 TO N
900:F2=F2+W(J)*J*X^(J-1)
910:NEXT J
920:RETURN
930:REM  * f'(0) *
940:F2=W(1)
950:RETURN
960:REM  * Printout *
970:PRINT "f(x)=";
980:FOR I=1 TO M
990:V= LEN C$(I)
 1000:Z=Z+V
 1010:IF (5+Z)>39 GOSUB 1070: GOTO 1030
 1020:PRINT C$(I);
 1030:NEXT I
 1040:RETURN
 1050:O=O-( LEN C$(0)+ LEN S$(0)): WAIT
      : CURSOR O,3: PRINT S$(0):O=5:Q=0
      : CLS
 1060:RETURN
 1070:LPRINT : LPRINT "      ";C$(I):Z=0
 1080:RETURN
 1090:Q=Q+1:O=5: CURSOR O,Q: PRINT C$(0
      ):O=O+ LEN C$(0)
 1100:RETURN
 1110:CLS : BEEP 1: CURSOR 2,1: PAUSE "
      Input Error !"
 1120:RETURN
```

2038 bytes

■ MEMORY CONTENTS

B$	data of "+", "–"
E	allowed value deviation
I	√
J	√
K	counter of answer
L	counter of cursors
M	√
N	maximum degree number
O	√
Q	√
V	√
X	X_n, X_{n+1}
Y$	work for selection
Z	√
X0	starting point
E1	accuracy for focusing
F1	$f(x_n)$
F2	$f'(x_n)$
I1	√
W(N)	√
X(N)	√
C$(N+1)	√
F$(0)	√
S$(0)	√

Program Title: **LINEAR REGRESSION PLOT**

Various statistical data can be processed accurately and swiftly!

Using this program, covariances, correlation coefficients and linear regression formulas, etc., between two related data (X_1, Y_1) and . . . (X_n, Y_n) can be obtained.

Results are displayed in both numerical values and graphs without any problems like complicated calculations and in drawing difficult graphs.

• HOW TO OPERATE

1. $\boxed{\text{R}}$ $\boxed{\text{U}}$ $\boxed{\text{N}}$ $\boxed{\text{ENTER}}$ [Program starts.]
   In accordance with the display, input each data item of (X_i, Y_i) in order. If $\boxed{\text{ENTER}}$ is pressed, you can shift to control 2 before the data (X_i) is input.

2. Select whether the data should be modified or not.
   - Operate $\boxed{\text{Y}}$ $\boxed{\text{ENTER}}$ if you want to check the data, then the data you have input will be displayed on the screen.
   - Operate $\boxed{\text{N}}$ $\boxed{\text{ENTER}}$ if the data is not to be checked. Then covariances, coefficients, regressions and average values are calculated and the results are output to the printer.

3. After the output of the above results, the Y value against the presumed value of X will be obtained. When the presumed value of X is input, the Y value against the X value will be calculated. The presumed value of X is ready to input. Then operate $\boxed{\text{ENTER}}$ only. Program ends.

   Note: Input capacity of data.
   - Standard capacity . . . . up to 8
   - When attached with CE-210M, CE-202M . . . . up to 255
     (However, value of F = $\underline{8}$ on line 10 should be changed.)

• CONTENTS

$$S_{xx} = \Sigma x_i^2 - n\bar{x}^2$$

$$S_{xy} = \Sigma x_i y_i - n\bar{x}\bar{y}$$

$$S_{yy} = \Sigma y_i^2 - n\bar{y}^2$$

$C = S_{xy}/(n-1)$ . . . . . . covariance

$r = S_{xy}/\sqrt{S_{xx}S_{yy}}$ . . . . correlation coefficient

$a = S_{xy}/S_{xx}$ regression coefficient ($y = ax + b$)

$b = \bar{y} - a\bar{x}$

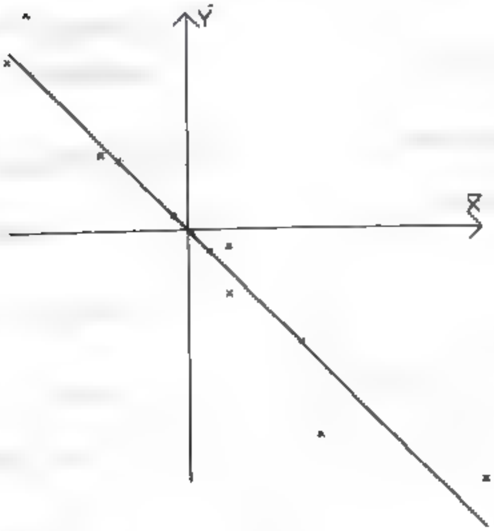
■ EXAMPLE

X	6.9	7.6	7.6	9.0	8.1	6.5	6.4	6.9
Y	12	10	9	5	6	15	14	12

```
covariance = –3.060714286
correlation coefficient = –9.693968513E–01
regression coefficient a = –3.942042318
                       b = 39.4475621
average value X̄ = 7.375
              Ȳ = 10.375
presumed value   X = 7,    Y = 11.8532
                 X = 8,    Y = 7.9112
                 X = 7.5,  Y = 9.8822
                 X = 7.3,  Y = 10.6706
                 X = 7.4,  Y = 10.2764
```

■ PRINTOUT (Colored Refer to page 4)

```
 * Linear Regression *
X(1)=6.9
Y(1)=12
X(2)=7.6
Y(2)=10
X(3)=7.6
Y(3)=9
X(4)=9
Y(4)=5
X(5)=8.1
Y(5)=6
X(6)=6.5
Y(6)=15
X(7)=6.4
Y(7)=14
X(8)=6.9
Y(8)=12

Covariance    =-3.060714286
Correlation   =-9.693968513E-01
* Regression Coefficient *
   A = -3.942042318
   B = 39.4475621
* Average value *
   X = 7.375
   Y = 10.375
```



```
* Presumption *
X =      7.0000    Y =     11.8532
X =      8.0000    Y =      7.9112
X =      7.5000    Y =      9.8822
X =      7.3000    Y =     10.6706
X =      7.4000    Y =     10.2764
```

■ KEY OPERATION SEQUENCE

1. R U N ENTER
[Program starts]

```
   * Linear Regression/Plot *
X (1) =   ?
```

2. 7 ENTER [Data of X_1 input]

```
   * Linear Regression/Plot *
X (1) =   7
Y (1) =   ?
```

3. 12 ENTER [Data of Y_1 input]

```
   * Linear Regression/Plot *
X (2) =   ?
```

⋮

Input the following data as above.

⋮

4. 12 ENTER [Input data of Y_8]

```
   * Linear Regression/Plot *

Confirm or Amend (Y/N) _
```

5. Y ENTER [Confirm data]

```
   * Linear Regression/Plot *
X (1) = 7
X (1) =   ?
```

6. 6.9 ENTER [Input correct data]

```
   * Linear Regression/Plot *
Y (1) = 12
Y (1) =   ?
```

⋮

Confirm data as above.

⋮

7. ENTER

```
   * Linear Regression/Plot *

Confirm or Amend (Y/N) _
```

8. N ENTER

```
   * Linear Regression/Plot *

          * Printout *
```

```
   * Linear Regression/Plot *

Presumed value X = _
```

9. 7 ENTER

```
   * Linear Regression/Plot *

Presumed value X = _
```

⋮

Repeat the following operations as above

⋮

10. 7.4 ENTER

```
   * Linear Regression/Plot *

Presumed value X = _
```

11. ENTER

```
   * Linear Regression/Plot *

      * Printout the value *
```

```

>
```

(When presumed value is displayed, the program ends.)

■ PROGRAM LIST

```
10:CLEAR : CLS : WAIT 0:F=8
20:DIM X(F),Y(F)
30:N=0: LPRINT CHR$ 27;"0": LPRINT
   CHR$ 27;"a": LPRINT CHR$ 13
40:CONSOLE 39: LPRINT
50:FOR B=1 TO F
60:CLS : WAIT 1: CURSOR 0,0: PRINT "*L
   inear Regression/Plot*"
70:BEEP 1: CURSOR 0,1: PRINT "X(";
   STR$ B;")=": CURSOR 7,1:G$="":
   INPUT G$
80:IF LEN G$=0 LET B=F: NEXT B: GOTO 1
   50
90:X(B-1)= VAL G$
100:CURSOR 0,2: PRINT "Y("; STR$ B;")='
   : CURSOR 7,2:G$="": INPUT G$
110:IF LEN G$=0 GOTO 100
120:Y(B-1)= VAL G$
130:N=N+1
140:NEXT B
150:IF N<3 WAIT 200: CLS : CURSOR 5,0:
   PRINT "** No Data **": GOTO 10
160:CLS : WAIT 1: CURSOR 0,0: PRINT "*L
   inear Regression/Plot*"
170:CURSOR 0,2: INPUT "Confirm or Amend
    (Y/N)";U$
180:IF U$="N" OR U$='n" GOTO 210
190:IF U$='Y" OR U$="y" GOSUB 1070
200:GOTO 160
210:CLS : CURSOR 0,0: PRINT "*Linear Re
    gress on/Plot*"
220:CJRSOR 2,2: PRINT "* Under Process
    *"
230:I-0:J=0:K=0:L=0:M-0
240:P=10^(98):O=-P:R=P:Q=0
250:FOR B=1 TO N
260:Z=B-1
270:I=I+X(Z)
280:J=J+Y(Z)
290:K=K+X(Z)*X(Z)
300:L=L+X(Z)*Y(Z)
310:M=M+Y(Z)*Y(Z)
320:IF P>X(Z) LET P=X(Z)
330:IF O<X(Z) LET O=X(Z)
340:IF R>Y(Z) LET R=Y(Z)
350:IF Q<Y(Z) LET Q=Y(Z)
360:NEXT B
370:I=I/N:J=J/N
380:K=K-N*I*I
390:L=L-N*I*J
400:M=M-N*J*J
410:H= SQR (K*M)
420:H=L/H
430:CLS : CURSOR 0,0: PRINT "*Linear Re
    gression/Plot*"
440:CURSOR 2,2: PRINT "* Printout *"
450:GOSUB 1210
```

```
460:LPRINT "Covariance   =";L/(N-1):
    LPRINT "Correlation  =";H
470:LPRINT "* Regression Coefficient *"
480:S=L/K:T=J-S*I
490:LPRINT "  A = ";S
500:LPRINT "  B = ";T
510:LPRINT "* Average value *"
520:LPRINT "  X = ";I
530:LPRINT "  Y = ";J
540:LPRINT CHR$ 27;"b": LPRINT "L0":
    LPRINT "M";200;",";-200: LPRINT "I'
550:A=(O-P)/350
560:B=(Q-R)/350
570:C=(I-P)/A
580:D=(R-J)/B
590:LPRINT "M";C;',";D
600:X1=-(I-P)/A:Y1=0
610:X2=(O-I)/A:Y2=0
620:GOSUB 1030
630:LPRINT "M";X2-10;",";Y2-10: LPRINT
     "D';X2;',';Y2
640:LPRINT 'D';X2-10;',';Y2+10
650:LPRINT 'PX'
660:LPRINT "M";X2-10;",";Y2+23: LPRINT
     "D";X2;",';Y2+23
670:X1=0:Y1=-(J-R)/B
680:X2=0:Y2=(Q-J)/B
690:GOSUB 1030
700:LPRINT "M";X2-10;",";Y2-10: LPRINT
     "D ;X2;",";Y2
710:LPRINT "D';X2+10;",";Y2-10
720:LPRINT "PY"
730:LPRINT "M';X2+10;",";Y2+5: LPRINT "
     D";X2+20;',";Y2+5: LPRINT CHR$ 27;
     1"
740:FOR E=1 TO N
750:X=(X(E-1)-I)/A:Y=(Y(E-1)-J)/B
760:GOSUB 1050
770:NEXT E
780:X1=-(I-P)/A:Y1=((S*P+T)-J)/B
790:X2=(O-I)/A:Y2=((S*O+T)-J)/B
800:LPRINT CHR$ 27; 2'
810:GOSUB 1030
820:N=1: LPRINT CHR$ 27;"3"
830:CLS : CURSOR 0,0: PRINT "*Linear Re
     gression/Plot*"
840:CURSOR 1,2:U$="": INPUT "Presumed v
     alue X=";U$:X(N-1)= VAL U$
850:IF U$="" GOTO 910
860:Y(N-1)=S*X(N-1)+T
870:X=(X(N-1)-I)/A:Y=(Y(N-1)-J)/B
880:LPRINT "M";X+2;",";Y: LPRINT "D";X-
     2;",";Y: LPRINT "M";X;",";Y-2:
     LPRINT "D";X;",";Y+2
890:N=N+1: IF N-1>F WAIT 200: CLS :
     CURSOR 3,1: PRINT " ** Presume End
     **": WAIT 0: GOTO 910
```

```
900:GOTO 830
910:CLS : CURSOR 0,0: PRINT "*Linear Re
    gression/Plot*"
920:CURSOR 0,2: PRINT "* Printout the v
    alue *"
930:LPRINT "M";-(I-P)/A;",";-(J-R)/B-20
940:IF N=1 THEN 1020
950:LPRINT CHR$ 27;"a"
960:LPRINT CHR$ 27;"0": LPRINT "* Presu
    mPtion *"
970:FOR W=1 TO N-1
980:USING "&&&&#####.####": LPRINT "X =
     ";X(W-1);
990:USING : LPRINT "    ";
 1000:USING "&&&&#####.####": LPRINT "Y
       = ";Y(W-1)
 1010:NEXT W
 1020:LPRINT : LPRINT : CLS : END
 1030:LPRINT "M";X1;",";Y1: LPRINT "D";
      X2;",";Y2
 1040:RETURN
 1050:LPRINT "M";X+2;",";Y+2: LPRINT "D
      ";X-2;",";Y-2: LPRINT "M";X-2;","
      ;Y+2: LPRINT "D";X+2;",";Y-2
 1060:RETURN
 1070:WAIT 0: FOR B=1 TO F
 1080:U$= STR$ X(B-1)
 1090:IF U$="0" GOTO 1200
 1100:CLS : CURSOR 0,0: PRINT "*Linear
      Regression/Plot*"
 1110:BEEP 1: CURSOR 0,1: PRINT "X(";
      STR$ B;")=";U$
 1120:CURSOR 0,2: PRINT "X("; STR$ B;")
      =":G$="": CURSOR 7,2: INPUT G$
 1130:IF LEN G$>0 THEN LET X(B-1)= VAL
      G$
 1140:U$= STR$ Y(B-1)
 1150:CLS : CURSOR 0,0: PRINT "*Linear
      Regression/Plot*"
 1160:BEEP 1: CURSOR 0,1: PRINT "Y(";
      STR$ B;")=";U$
 1170:CURSOR 0,2: PRINT "Y("; STR$ B;")
      =":G$="": CURSOR 7,2: INPUT G$
 1180:IF LEN G$>0 THEN LET Y(B-1)= VAL
      G$
 1190:NEXT B
 1200:RETURN
 1210:LPRINT " * Linear Regression *":
      FOR B=1 TO F
 1220:U$= STR$ X(B-1)
 1230:IF U$='0" GOTO 1280
 1240:LPRINT "X("; STR$ B;")-";U$
 1250:J$= STR$ Y(B-1)
 1260:LPRINT "Y("; STR$ B;")=";U$
 1270:NEXT B
 1280:LPRINT
 1290:RETURN
```

2794 bytes

■ MEMORY CONTENTS

A	graph coefficient (X)
B	graph coefficient (Y)
C	√
D	√
E	√
F	number of data
G$	area for input
H	correlation coefficient
I	$\bar{X}$
J	$\bar{Y}$
K	Sxx
L	Sxy
M	Syy
N	√
O	max. (X)
P	min. (X)
Q	max. (Y)
R	min. (Y)
S	regression coefficient "a"
T	regression coefficient "b"
U$	√
W	√
X	√
Y	√
Z	√
X1	for line drawing sub-routine
X2	for line drawing sub-routine
Y1	for line drawing sub-routine
Y2	for line drawing sub-routine
X(F)	storage of input and presumed data
Y(F)	storage of input and presumed data

BUSINESS SOFTWARE

The principal functions of the business software of the PC-2500 are listed below.

(1) It lets you create tables quickly and easily.
(2) By specifying a formula, it performs table calculations automatically upon data entry. And if any of the data is changed, it immediately recalculates the table.
(3) It lets you display the data in the table in graph form.
(4) It lets you store names and telephone numbers so that you can use the PC-2500 as a personal telephone directory.

The software also has many other functions, but now we will just describe the graph function.

"Tables," particularly number tables, are used often in business. We use the numbers, or rather data, in number tables to make various judgements and forecasts.

Take the following table for example.

Transition in Sales

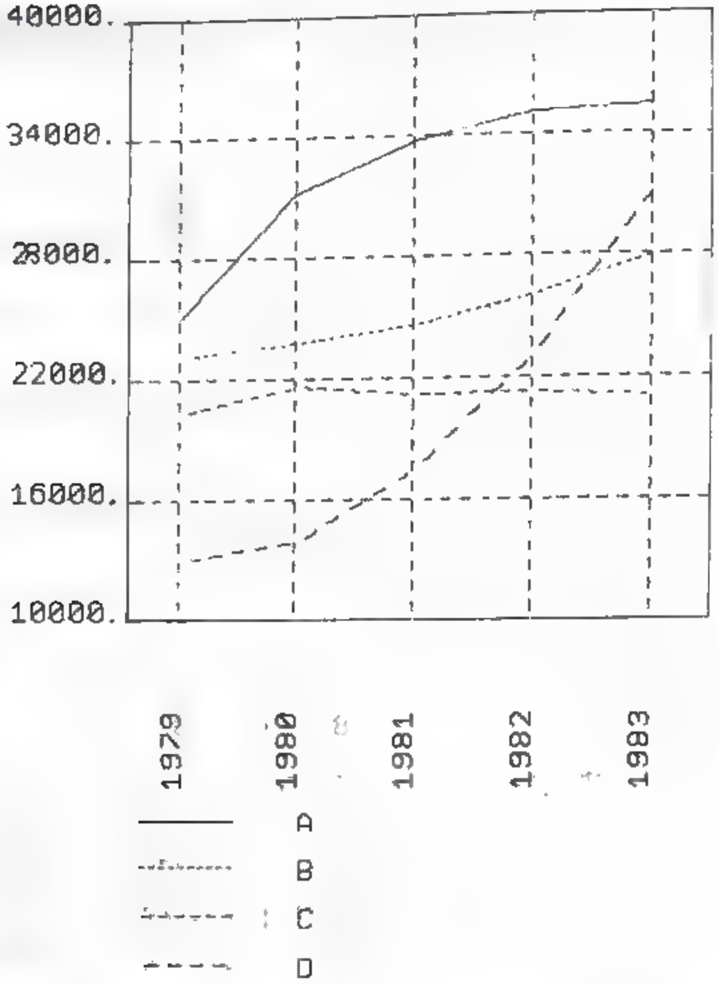
COMPANY YEAR	A	B	C	D	TOTAL
1979	25,000	23,150	20,332	12,926	81,408
1980	31,250	23,828	21,649	13,956	90,683
1981	33,750	24,630	21,223	17,446	97,049
1982	35,100	26,012	21,254	23,028	105,394
1983	35,421	27,962	21,059	31,087	115,529

This table shows the transition in sales of a certain product by manufacturer. Let's look at some examples of what we can discover from this table.

- Company A maintained top sales during the period.
- However, the sales of company D are catching up fast.
- The rapid growth of the sales of company D indicates that it will just be a matter of time before company D takes the lead away from company A.
- Therefore, company A must quickly devise a plan to maintain its lead.
- The sales of company B are also gradually growing.

However, to be able to immediately see these trends from just looking at the table is extremely difficult, and those people who can are either very good with numbers or can read tables very well. Chances are you will not be working with people like that very often.

Let's see what the table above looks like in the form of a graph.



This graph tells us at a glance that company A is top in sales, that the sales of company D are growing rapidly, and that the sales of company B are growing steadily. It is also quite clear that company D will probably overtake company A and that sometime in the future company B, too, will threaten company A's position.

By transforming a number table into a graph in this way, problem areas and trends are clarified, thereby greatly assisting judgment of a situation and decisions. Also, the use of graphs to clarify explanations in papers and reports helps the person reading the material to understand the situation better.

Graphs can be extremely helpful, but they can require a lot of time and effort to make. By using the software of the PC-2500, graphs can be made quickly and easily, so you will be free to use them much more often.

1. OUTLINE OF BUSINESS SOFTWARE

Simply stated, the business software for the PC-2500 is designed for creating tables, performing table calculations, and creating graphs based on the resultant data through a simple operation.

For example, in order to make a table for the transition in sales described in the previous section, the following procedure is used.

(1) Enter the item names and equation in the PC-2500. The item names include the years, the names of the companies, and "Total" and are used as the heading for each item. The equation is used to obtain the total.

(2) When the item names and equation have been entered, the table is created automatically, so just enter the data in order.

(3) When one row of data has been entered, it is automatically calculated. In the table talked about in the previous section, when the data for each company for 1979 has been input, that row of data is automatically totalled and the result entered in the total column.

(4) When the data for each year has been input, it is calculated in the same way. If, later, you discover that you entered incorrect data, you can simply replace that data with the correct data, and it will be recalculated immediately.

(5) Depending on the table or graph, you may want to rearrange the data in order from greatest to smallest or vice versa. Also, you may want to uniform the number of decimal places in the data or calculation results. This software lets you specify these types of operations on the data.

(6) You can also print out the tables you've created for use in reports, etc.

(7) Furthermore, by following the messages given by the PC-2500, you can make the following four types of graphs by just selecting the items and conditions.
Bar graph
Line graph
Band graph
Pie graph

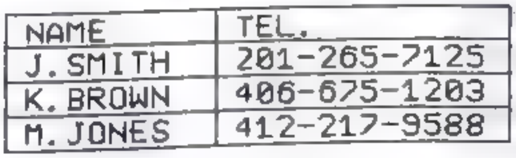
(8) By saving the tables you've made on tape, you can use them anytime you need them to make graphs or change the data.

(9) You can also store your tables on RAM cards, sold separately.

This is the procedure used to create tables and utilize data.

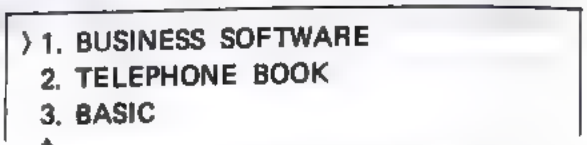
Telephone List

Item headings for a telephone list have already been set in the PC-2500, so you can make your own telephone list by just entering people's names and their telephone numbers.



NAME	TEL.
J.SMITH	201-265-7125
K.BROWN	406-675-1203
M.JONES	412-217-9588

Let's look at a simple example here. First, try making the telephone list show above. Turn on the power, and the menu shown at the right will appear.



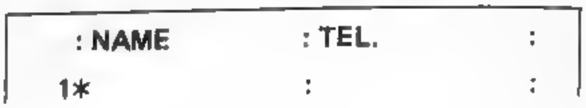
```
> 1. BUSINESS SOFTWARE
  2. TELEPHONE BOOK
  3. BASIC
```

Any of these functions can be selected by inputting their respective numbers from the number keys.
They can also be selected by moving the index with the [↑] or [↓] key and then pressing the [ENTER] key.

- To display this menu from another display, press the [SHIFT] + [illegible] keys. (Press the [illegible] key while holding down the [SHIFT] key.)

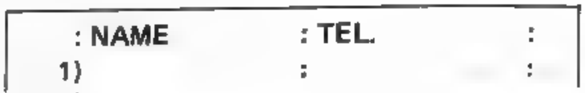
Then press the [2] key to select "TELEPHONE BOOK" from the menu.

When used for the first time, "**INITIALIZING **" will be displayed before the display at the appears.



```
   : NAME          : TEL.          :
 1*                :               :
```

Next, specify "WRITE" by pressing the [DEF] and [↑] keys.

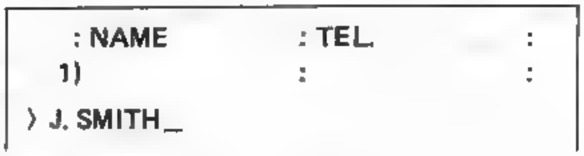


```
   : NAME          : TEL.          :
 1)                :               :
```

Index
Indicates that data can be entered in this column.

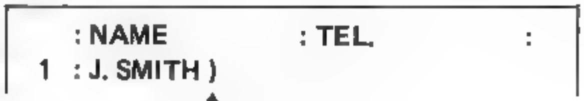
Now you are ready to enter a name.

J. SMITH



```
   : NAME          : TEL.          :
 1)                :               :
> J. SMITH_
```

Press the [ENTER] key, and in a few moments the name you input is entered under the item "NAME" and the index moves to the "TEL" item. (You can move the index to the next line to enter another name.)

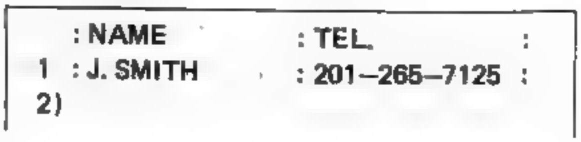


```
   : NAME          : TEL.          :
 1 : J. SMITH )
```

Index

Next enter the telephone number.

201–265–7125
[ENTER]

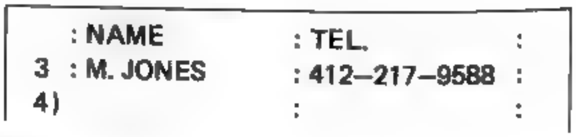


```
   : NAME          : TEL.          :
 1 : J. SMITH      : 201–265–7125  :
 2)
```

After a name and telephone number have been input in this manner, the index then moves to the next row.

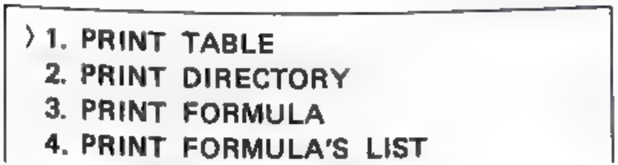
Use the same procedure to enter other names and telephone numbers.

K. BROWN	[ENTER]
406–675–1203	[ENTER]
M. JONES	[ENTER]
412–217–9588	[ENTER]



```
   : NAME          : TEL.          :
 3 : M. JONES      : 412–217–9588  :
 4)                :               :
```

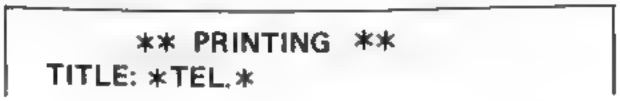
When you are finished entering, print out the telephone list.
Press the DEF and PRINT CLS keys.



```
〉1. PRINT TABLE
 2. PRINT DIRECTORY
 3. PRINT FORMULA
 4. PRINT FORMULA'S LIST
```

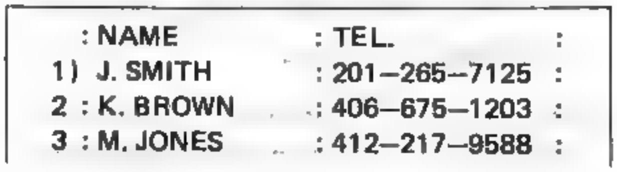
Each of these functions can be executed by inputting their respective numbers from the number keys.
They can also be executed by moving the index with the ↓ or ↑ key and then pressing the ENTER key.

To print the data (names, telephone numbers), now in the form of a table, press the 1 key, and printing will begin.



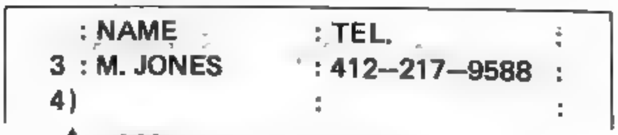
```
      ** PRINTING **
TITLE: *TEL.*
```

An empty row is printed at the bottom because when a telephone number is entered, the next row is automatically made. To delete it, use the following procedure.
Press the DEF and WRITE ↑ keys to set the data write mode.



```
   : NAME          : TEL.           :
1) J. SMITH        : 201–265–7125   :
2 : K. BROWN       : 406–675–1203   :
3 : M. JONES       : 412–217–9588   :
```

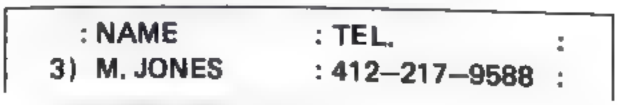
After checking that the index is displayed, press the ↓ (or SHIFT + ↓ keys) to move the index to the bottom row.



```
   : NAME          : TEL.           :
3 : M. JONES       : 412–217–9588   :
4)                 :                :
```

Index moves to this position.

Press the SHIFT + DEL BS keys, and the fourth row (bottom row) will be deleted.



```
   : NAME          : TEL.           :
3) M. JONES        : 412–217–9588   :
```

Next, let's try a table calculation. A detailed explanation is given later, so first just operate the computer by following the directions.

The table on the next page shows the sales for three stores by product. Use this data to make a table and graph and then print them out.

(Example)

Product	A_STORE	B_STORE	C_STORE	TOTAL
TV	1240	1890	1048	
VIDEO	980	1350	870	
AUDIO	1428	864	1250	

First, press the [SHIFT] + [ON/BRK] keys.

```
) 1. BUSINESS SOFTWARE
  2. TELEPHONE BOOK
  3. BASIC
```

Next, press the [1] key.
Wait a few moments and the display at the right will appear. If another display (*TEL.*, etc.) appears, press the [DEF] and [PEN] PROGRAM keys. The display shown will appear.

```
     ** ENTER FORMULA **
)_
```

Next, enter the "formula." A formula specifies the item names and equation.

Enter the data as shown below.
"PRODUCT": A_STORE: B_STORE: C_STORE:
TOTAL = A_STORE + B_STORE + C_STORE

Note: As shown by "Total = A_STORE + B_STORE + C_STORE," the equation places the result of the calculation on the left of the equal sign and the formula on the right side of the equal sign.

```
     ** ENTER FORMULA **
)
```

Check to be sure it is entered correctly. If it is correct, press the [ENTER] key. This will cause the table to be created.

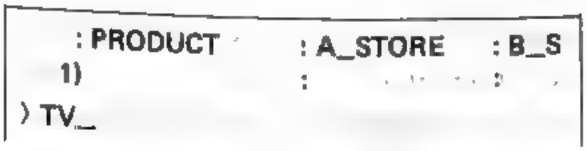
```
     ** TABLE PROCESSING **
```

```
   :PRODUCT       : A_STORE    : B_S
  1)              :            :
  ↑
```

This is the index. This index indicates an item under which characters are input. (Enter characters under this item.)

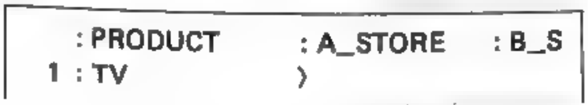
Now let's enter the data. First, enter the data "TV" for the first row.

TV



```
   : PRODUCT        : A_STORE    : B_S
 1)                 :            :
) TV_
```

ENTER

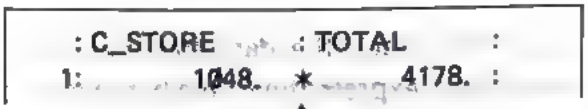


```
   : PRODUCT        : A_STORE    : B_S
 1 : TV             )
```

This type of index indicates that numbers are entered under this item.

Continue by entering the data for store A, store B and store C.

1240 ENTER
1890 ENTER
1048 ENTER

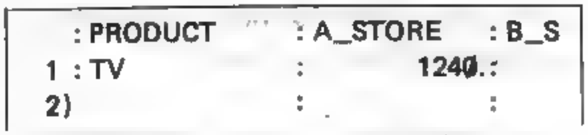


```
   : C_STORE        : TOTAL         :
 1:          1Ø48.  *       4178.   :
```

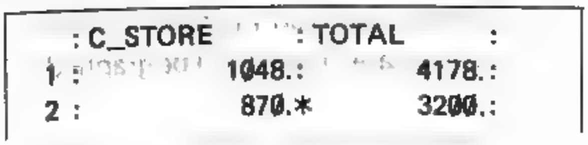
This symbol is also an index and indicates an item under which calculation results are entered. Therefore, data cannot be input in this column.

When the data for store C is input, the total amount for the first row (TV) is calculated, and the result is entered in the "Total" column.
To proceed with the second row, first press the ENTER key, and the columns in the second row are created. Then continue by entering each of the respective data.

VIDEO ENTER
980 ENTER
1350 ENTER
870 ENTER



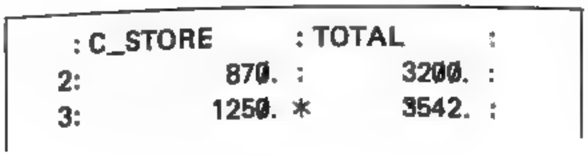
```
   : PRODUCT        : A_STORE    : B_S
 1 : TV             :       124Ø.:
 2)                 :            :
```



```
   : C_STORE        : TOTAL         :
 1 :          1Ø48.:         4178.:
 2 :           87Ø.*         32ØØ.:
```

Enter the data for the third row in the same way.

ENTER
AUDIO ENTER
1428 ENTER
864 ENTER
1250 ENTER



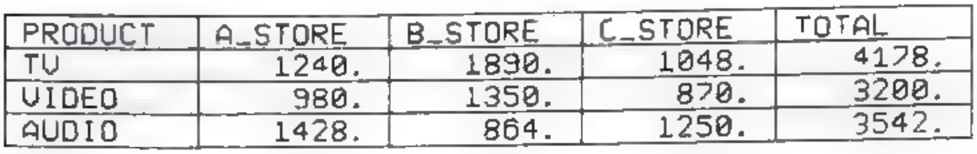
```
   : C_STORE        : TOTAL         :
 2:           87Ø. :         32ØØ. :
 3:          125Ø. *         3542. :
```

When you have completed entering the data, print out the table.
Press the DEF and CLS PRINT keys and then the 1 key; the display at the right will appear and the table will be printed.

```
1. PRINT TABLE
2. PRINT DIRECTORY
3. PRINT FORMULA
4. PRINT FORMULA'S LIST
```

```
   ** PRINTING **
```

(Printing sample)



PRODUCT	A_STORE	B_STORE	C_STORE	TOTAL
TV	1240.	1890.	1048.	4178.
VIDEO	980.	1350.	870.	3200.
AUDIO	1428.	864.	1250.	3542.

Now let's stack the data for stores A and B and express them in a bar graph.
Press the DEF and MODE keys.

Stack: To include the data from several items in one graph.

```
〉1. BAR
 2. LINE
 3. BAND
 4. PIE
```

Then press the 1 key to select a bar graph.

```
          SELECT COLUMN
〉PRODUC       A_STOR       B_STOR
 C_STOR       TOTAL
```

Next, specify the items to be entered in the graph. First, specify store A by moving the index to the "A–STORE" position with the → key.

```
          SELECT COLUMN
 PRODUC     〉A_STOR       B_STOR
 C_STOR     ↑ TOTAL
            └─ index
```

Then press the * key. An asterisk (*) will be displayed in front of store A. This indicates that it will be graphed.

```
          SELECT COLUMN
 PRODUC    〉*A_STOR      〉B_STOR
 C_STOR     ↑TOTAL
            └─* is displayed.
```

Next, specify store B. Move the index to the "B–STORE" position with the → key.

```
          SELECT COLUMN
 PRODUC    *A_STOR     〉B_STOR
*C_STOR    TOTAL
```

Then press * ENTER . (The * key can be omitted.)
Store B is specified and the display on the right appears.

```
ANY OTHER COLUMNS?
  1. YES
〉2. NO
```

- If there is still an item to be specified, press the 1 key.
  The display shown above to specify items reappears.
- After specifying the items, press the 2 key or ENTER key.

Since we have already specified stores A and B and there are no other items to be stacked, press the 2 key.

```
FRAME            MAX = A
NO GRID          MIN  = A
〉1. DRAW GRAPH
  2. SET GRAPH FORMAT
```

The top two lines of the information displayed are the conditions for drawing the graph. To draw a graph with these conditions, press the 1 key.
For this graph you will change the conditions, so press the 2 key.

```
〉1. FRAME   (DRAW QUICKLY)
  2. HATCH   (W/BLACK HATCH)
  3. COLOR   (W/COLOR HATCH)
```

To specify color, press the 3 key.

```
DRAW WITH GRID?
〉1. NO GRID (SCALE ONLY)
  2. SCALE AND GRID
```

To specify a grid, press the 2 key.

```
SET MAX. AND MIN. VALUE
MAX. = A_
```

Next, specify the maximum and minimum values for the scale. Set the maximum value to automatic (the computer automatically sets the value).
Press ENTER.

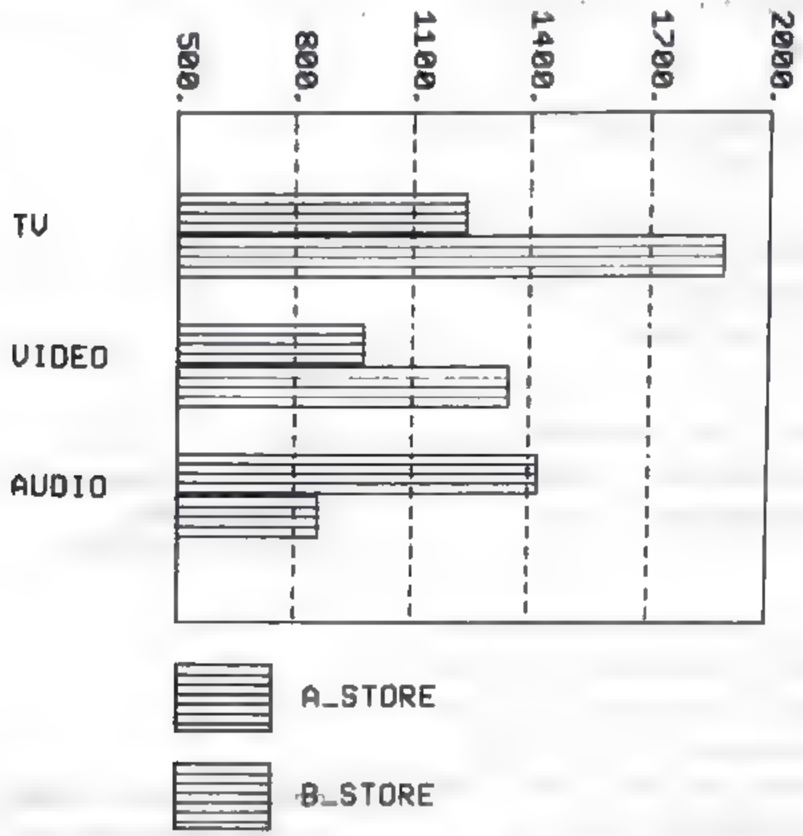
Set the minimum value to 500.
Press the CLS key, then enter 500 (500 ENTER).

```
SET MAX. AND MID. VALUE
MAX. = A
MIN. = 500
```

```
   ** DRAWING **
```

This completes setting of the conditions. The graph is then printed.

(**Printing sample**)

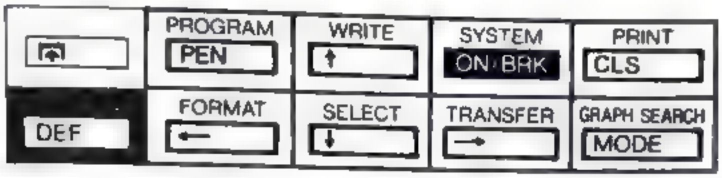


(If you want to print words other than those specified as the items for columns, enter the words following a colon (:) when specifying the items to be printed.)

The above is a general description of how to make a table and graph. It is the general procedure for creating tables and graphs, but there are also various other functions that will be explained on the following pages.

2. BUSINESS SOFTWARE KEYS

The business software has many various functions, and a key to execute each function has been provided. Below is a description of these keys and the functions that can be selected with them.



Keys	Basic or example display	Description
SHIFT + ON/BRK	〉1. BUSINESS SOFTWARE 2. TELEPHONE BOOK 3. BASIC	This display lets you select the following functions. 1 key : BUSINESS SOFTWARE (Permits use of the business software.) 2 key : TELEPHONE LIST (Calls up the telephone list.) 3 key : BASIC (Permits use of BASIC.) ● To switch from the business software to BASIC or vice versa, it must be done from this display. The telephone list is a function (table) of the business software.
DEF SYSTEM ON/BRK	〉1. DISPLAY FREE AREA 2. DELETE FILE 3. DELETE ALL FILES 4. USE RAM CARD	1 key : Used to check the memory status and the amount of free memory remaining. 2 key : Used to delete the specified table or specify a table and delete it. 3 key : Used to delete all stored tables and data. 4 key : Used to select how the RAM card (optional), when attached, is to be used.
DEF WRITE ↑	:NAME :TEL. : 1) : :	Used to call up the specified table to permit revision of, addition to, or deletion of the data.
DEF PROGRAM PEN	✱✱ ENTER FORMULA ✱✱ 〉_	Used to enter a new formula to create a table.
DEF PRINT CLS	〉1. PRINT TABLE 2. PRINT DIRECTORY 3. PRINT FORMULA 4. PRINT FORMULA'S LIST	1 key : Used to print the specified table. 2 key : Used to print all the titles of stored tables. 3 key : Used to print the formula of the specified table. 4 key : Used to print all the formulas of stored tables.

Keys	Basic or example display	Description
DEF FORMAT ←	〉1. SET TOTAL/AVERAGE ROW 2. CHANGE DECIMAL FORMAT 3. SORT BY COLUMN 4. DISPLAY FORMULA	1 key : Used to specify the addition of or delete the total and/or average row at the bottom of the table. 2 key : Used to specify a numerical display format. The following formats can be selected. ● Automatic: Normal display format. ● Business calculation: Permits specification of the number of decimal places. ● Scientific calculation: Displays in exponential notation and permits specification of the number of significant digits. 3 key : Permits sorting of the data in descending or ascending order. 4 key : Used to display or correct the formula for the specified table.
DEF SELECT ↓	〉✱TEL.✱ PRODUCT	Recalls the titles of stored tables. By using the →, ←, ↓ or ↑ key to move the index to select a title, data can be entered in that table, the display format specified, the table printed or a graph created.
DEF →	〉1. CASSETTE TAPE 2. SERIAL INTERFACE	1 key : By using a tape recorder, table data can be stored on or read from the tape or verified. 2 key : This function permits sending or receiving of table data by means of a serial input/output device.
DEF MODE	〉1. BAR 2. LINE 3. BAND 4. PIE	Enters the graph mode. 1 key : Creates a bar graph. 2 key : Creates a line graph. 3 key : Creates a band graph. 4 key : Creates a pie graph.
MODE	: NAME : TEL. 1 : J. Smith : Ø6–621–1221 2 : K. Brown : Ø7435–3–552 SEARCH : _	Used to search data in each of the items while the table is being displayed.

- Except for the following instances, a function is selected when the respective business software key is pressed after the DEF key.
  (1) When the menu that appears after pressing the SHIFT + ON/BRK keys is displayed
  (2) When using BASIC
  (3) While printing
  (4) While transferring a table
- The 1 , 2 , 3 and 4 keys are used to select their respective functions, but the desired function can also be selected by pressing the ENTER key when the function is indicated by the index, which can be moved with the ↑ or ↓ key, in each of the function selection displays.

3. DESCRIPTIONS OF EACH FUNCTION

3-1) Starting the Business Software

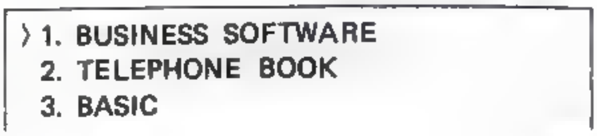
The display shown at the right will appear in the following conditions:

(1) When the power switch is turned on.

(2) When the [SHIFT] + [ON/BRK] keys are pressed.

(3) When the ALL-RESET switch is pressed. (See page 353.)

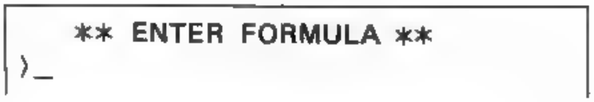
(4) When the LOW BATTERY lamp lights during printing.



```
〉1. BUSINESS SOFTWARE
  2. TELEPHONE BOOK
  3. BASIC
```

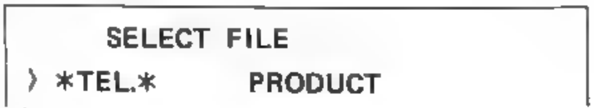
When the [1] (or [ENTER]) key is pressed in this display, the business software is started. The display when the business software is started will be as follows depending on whether or not there are any tables other than the telephone list.

(1) When there is no table other than the telephone list, the computer will enter the program mode. The first time the business software is started after deleting all of the tables, the display at the right will be shown.



```
      ** ENTER FORMULA **
〉_
```

(2) When there is a table or tables other than the telephone list, their titles will be shown. (Enters the select mode.)



```
      SELECT FILE
〉 *TEL.*      PRODUCT
```

Index

The index is shown at the position of the title for tables specified before this. When many tables have been stored and their titles cannot be shown on one screen, the titles will be displayed within the range that can be displayed on the screen.

Note: The telephone list is a table created with the business software.

Note: The following explanations assume that the business software has already been started.

The message "TOO MANY DATA" is displayed when the free memory area is not enough, then the initial screen is displayed.

3-2) Creating Formulas and Tables

The business program will enable you to create tables and graphs. Before doing so, however, you must write a "formula."

We will describe the "formula" and the "table."

Note: The message

MEMORY OVERFLOW

may be displayed while you are entering the "formula" or creating a table. This means that the memory within the PC-2500 used to store data is full and that no further data can be stored.

If this occurs, see page 351 and delete the previous table.

(Before you delete the table, save it on tape so that you can load and use it again.)

Do not delete the table "PRODUCT" that we created earlier since we will use it in the following explanations.

[1] What is a Formula?

A formula is something you specify and consists of the following:

- Item name
- Number of items
- Equation

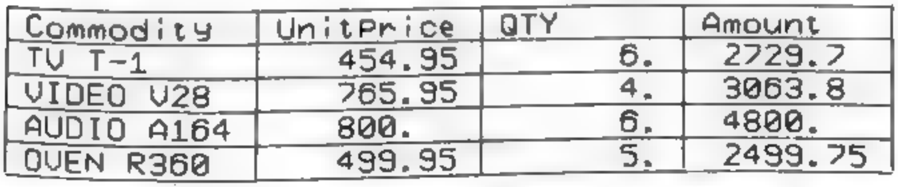
The formula specifies the above to create a table. It has the following form.

Examples: Formulas and Tables

(1) "Commodity": UNIT PRICE: QUANTITY: AMOUNT = UNIT PRICE* QUANTITY

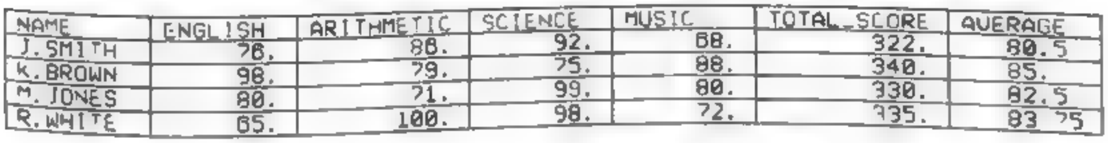
This formula can also be written with the item names omitted. The omitted item names will be taken from the equation.

"Commodity: AMOUNT = UNIT PRICE * QUANTITY



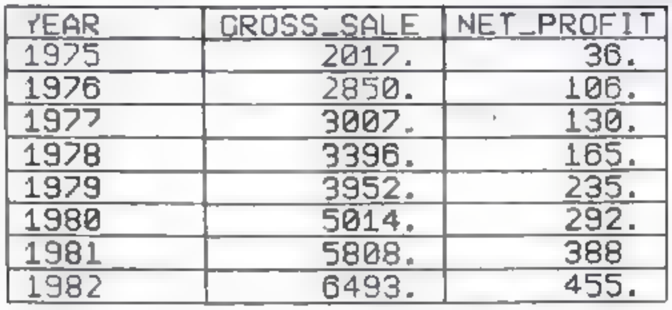
Commodity	UnitPrice	QTY	Amount
TV T-1	454.95	6.	2729.7
VIDEO V28	765.95	4.	3063.8
AUDIO A164	800.	6.	4800.
OVEN R360	499.95	5.	2499.75

(2) "NAME": TOTAL SCORE = ENGLISH + ARITHMETIC + SCIENCE + MUSIC: AVERAGE = TOTAL SCORE/4



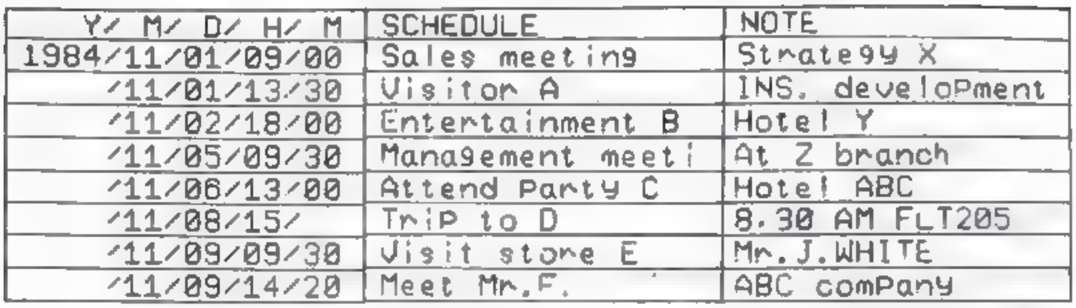
NAME	ENGLISH	ARITHMETIC	SCIENCE	MUSIC	TOTAL SCORE	AVERAGE
J.SMITH	78.	88.	92.	68.	322.	80.5
K.BROWN	98.	79.	75.	88.	340.	85.
M.JONES	80.	71.	99.	80.	330.	82.5
R.WHITE	65.	100.	98.	72.	335.	83.75

(3) "YEAR": GROSS SALE: NET PROFIT



YEAR	GROSS_SALE	NET_PROFIT
1975	2017.	36.
1976	2850.	106.
1977	3007.	130.
1978	3396.	165.
1979	3952.	235.
1980	5014.	292.
1981	5808.	388
1982	6493.	455.

(4) "Y/M/D/H/M": SCHEDULE: NOTE

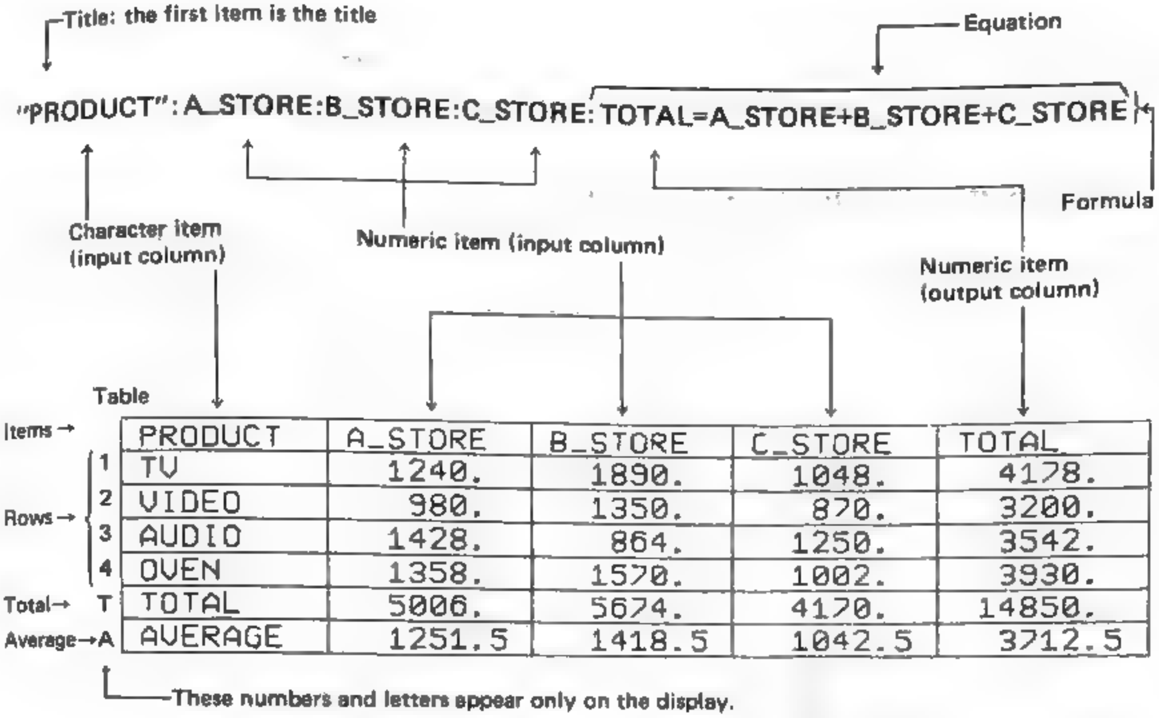


Y/ M/ D/ H/ M	SCHEDULE	NOTE
1984/11/01/09/00	Sales meeting	Strategy X
/11/01/13/30	Visitor A	INS. development
/11/02/18/00	Entertainment B	Hotel Y
/11/05/09/30	Management meeti	At Z branch
/11/06/13/00	Attend Party C	Hotel ABC
/11/08/15/	Trip to D	8.30 AM FLT205
/11/09/09/30	Visit store E	Mr.J.WHITE
/11/09/14/20	Meet Mr.F.	ABC company

As shown in the above examples, the item names, number of items, and equation are specified in the formula. A formula need not contain an equation, as shown in (3), or may consist only of character items, as shown in (4). (For convenience, these will also be called "formulas.")

Terminology

The terms, related to formulas and tables, used in the text are illustrated below.



"PRODUCT":A_STORE:B_STORE:C_STORE:TOTAL=A_STORE+B_STORE+C_STORE

	PRODUCT	A_STORE	B_STORE	C_STORE	TOTAL
1	TV	1240.	1890.	1048.	4178.
2	VIDEO	980.	1350.	870.	3200.
3	AUDIO	1428.	864.	1250.	3542.
4	OVEN	1358.	1570.	1002.	3930.
T	TOTAL	5006.	5674.	4170.	14850.
A	AVERAGE	1251.5	1418.5	1042.5	3712.5

Title: the first item is the title
Equation
Formula
Character item (input column)
Numeric item (input column)
Numeric item (output column)
Table
Items →
Rows →
Total →
Average →
These numbers and letters appear only on the display.

- When the formula, table, and the data contained in the table are shown together, they will be called, collectively, table data.

Note: If the optional RAM card is not used, a table of about 10 columns by 7 rows, or a telephone list for 80 persons can be created (see page 319). Use the optional RAM card if you plan to create a larger table or telephone list, or several tables. If you create a table or telephone list without the RAM card and then install the RAM card, the table or telephone list data will be cleared. Therefore, install the RAM card before creating the table or telephone list. Or, see page 346 and save the table or telephone list on tape, install the RAM card, and then load the table or telephone list from the tape.

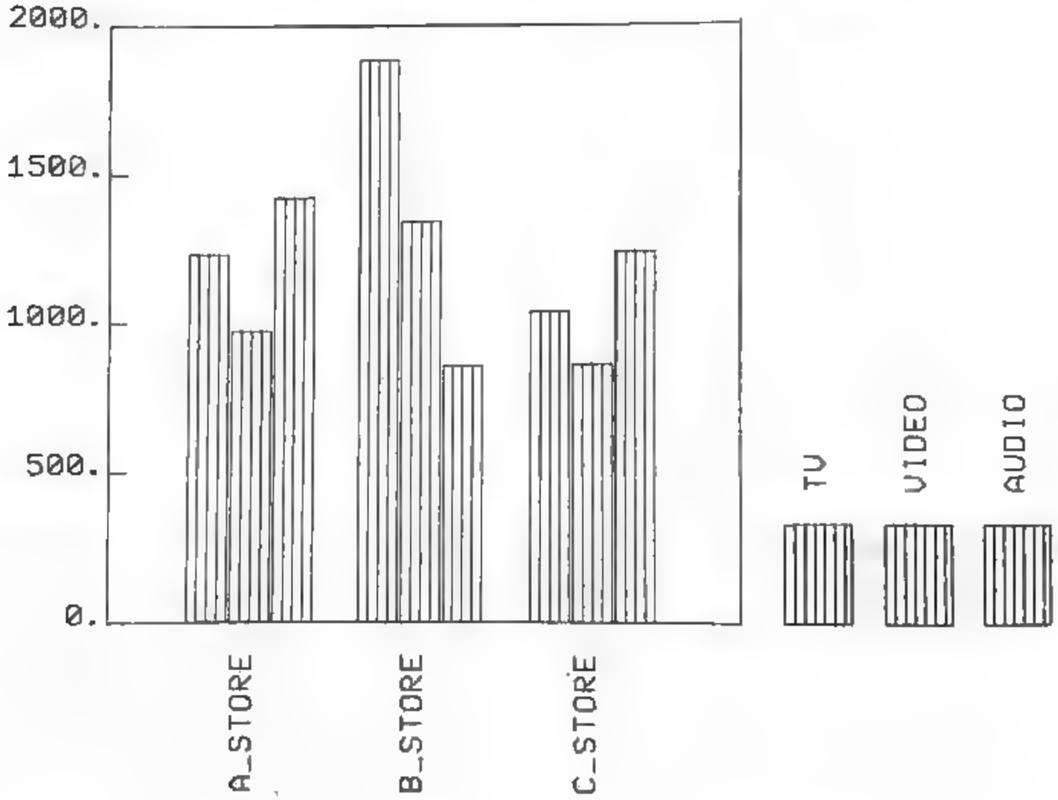
= Tables and Graphs =

To create a graph, we first create a table and then use its data.
Let's look at the following example.

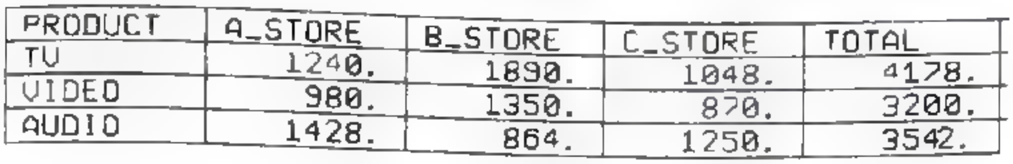
Example: Graphs in which the columns and rows are reversed.
(Compare this with the table and graph on the next page.)

⟨ 1. Sales at each store for various stores ⟩

PRODUCT	TV	VIDEO	AUDIO
A_STORE	1240.	980.	1428.
B_STORE	1890.	1350.	864.
C_STORE	1048.	870.	1250.
TOTAL	4178.	3200.	3542.

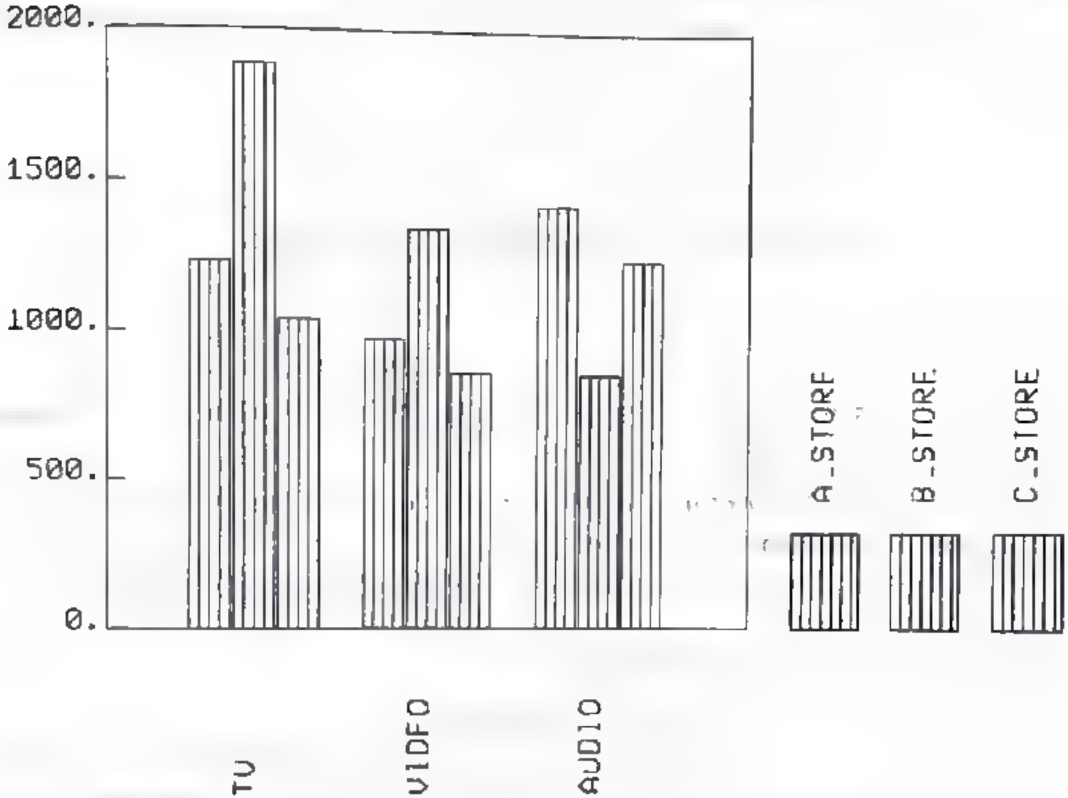


(2. Sales at each product for various products)



PRODUCT	A_STORE	B_STORE	C_STORE	TOTAL
TV	1240.	1890.	1048.	4178.
VIDEO	980.	1350.	870.	3200.
AUDIO	1428.	864.	1250.	3542.

Note: The columns and rows in graph 1 are reversed in this graph.



The PC-2500 creates a graph from the column data in the table. Decide on the kind of graph you want to have before creating the table.

[1] Writing a Formula

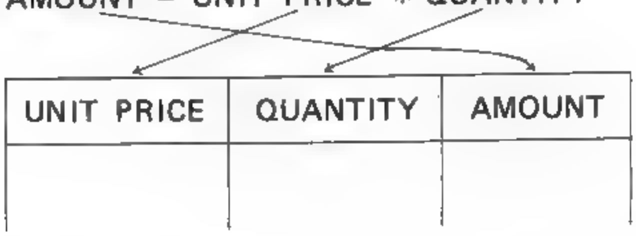
Note the following when writing a formula.

(1) Write the item expressing the result of the equation (item for the output column) on the right side of the equal sign (=).

Example: $\underbrace{\text{AMOUNT}}_{\text{Output column}} = \underbrace{\text{UNIT PRICE}}_{\text{Input column}} * \underbrace{\text{QUANTITY}}_{\text{Input column}}$

- If only the equation is written (without the item names) as shown above, the output column is positioned to the right of the input columns so that the result can be obtained after you input the data.

Example: AMOUNT = UNIT PRICE * QUANTITY



UNIT PRICE	QUANTITY	AMOUNT

(2) If the item is to contain characters (character item), enclose the item name in double quotation marks (" ").

(3) If the item is to contain numeric values (numeric item), be sure the first character of the item name is a letter.
The formula will be considered illegal if the first character is a symbol (e.g. #, &, %, etc.) or number.
The following cannot be used as item names.

- Names which contain function commands
(Names such as SIN, COS, TAN, INT, PI, etc., cannot be used. However, names such as "sin" (lower case letters) can be used for an item name.)

  Example: $\underbrace{\text{COS}}\text{T} = \ldots\ldots$

  "COS" will be taken to mean the trigonometric function cosine. Use lower case letters such as "Cost".

- Calculation commands or symbols which determine the order of a calculation such as +, —, *, /, ^, (,), etc.

(4) Separate the item names and equations with a colon (:).

(5) An equation must be one of the following types.
ⓐ Including character terms only.
ⓑ Including numeric terms only.
ⓒ Including one character term and numeric terms.

(7) Up to 20 items can be created. If the number of items exceeds 20 when the formula is being converted to the table, the following message appears:

```
        ** TABLE PROCESSING **
```

```
        ** TOO MANY COLUMNS **
```

Afterwards, the program again requests the input formula. Reduce the number of items to 20 or less.

(8) Up to 16 characters can be used as an item name in a table.
If you used more than 16 characters for an item name in the formula, only the first 16 characters are used in the table after the formula is converted.

(9) Up to 78 characters can be used (input) in the formula.

[3] Table Generation

Table generation in the business program starts once the formula is input. To enter the formula, press DEF PEN .
(If a table is not stored when you enter the business program, table generation immediately starts. See page 308.)

Press

DEF PEN

```
        ** ENTER FORMULA **
>_
```

and obtain the display on the right.
Let's use the following example in our explanation.

(Example)

STORE NAME	STAFF	RESULT	QUOTA	ATTAINMENT
A_STORE	32	4752	4500	
B_STORE	18	2870	2500	
C_STORE	27	4560	3800	
D_STORE	34	4682	4750	
E_STORE	40	5610	5600	
TOTAL				
AVERAGE				

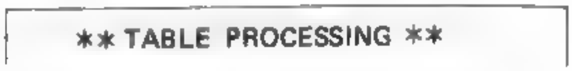
(Formula) "STORE NAME": STAFF: ATTAINMENT = RESULT/QUOTA * 100

(1) First, enter the formula.
"Store Name": Staff: ATTAINMENT = RESULT/QUOTA * 100

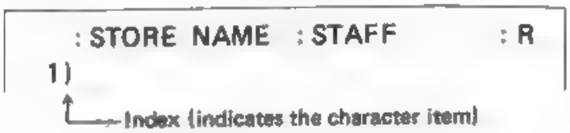
```
     ** ENTER FORMULA **
```

(2) Check to see that there are no mistakes in the formula.
If there is a mistake, correct it using the [↑], [↓], [→], [←], [INS], and [DEL BS] keys.
The formula can also be corrected in the CHECK FORMULA mode.

(3) Press the [ENTER] key. The formula will be converted to table form.



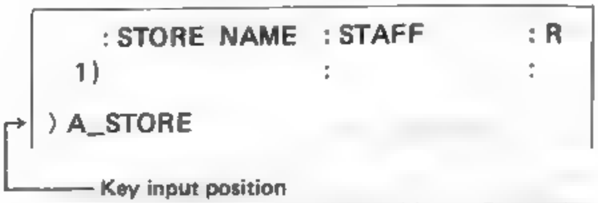
```
** TABLE PROCESSING **
```



```
   : STORE NAME  : STAFF        : R
 1)
```

Index (indicates the character item)

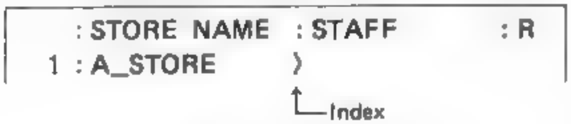
(4) Now enter the data.
A_STORE



```
   : STORE NAME  : STAFF        : R
 1)              :              :
) A_STORE
```

Key input position
Your key inputs are displayed here. By pressing the [ENTER] key, the key inputs are entered into the item indicated by the index.
(indicates a numeric item)

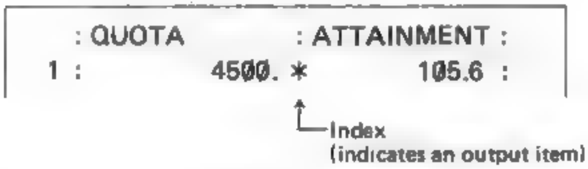
[ENTER]



```
   : STORE NAME  : STAFF        : R
 1 : A_STORE     )
```

Index

32 [ENTER]
4752 [ENTER]
4500 [ENTER]

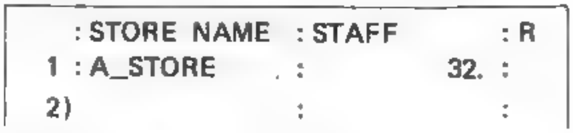


```
   : QUOTA         : ATTAINMENT :
 1 :        45ØØ. *       1Ø5.6 :
```

Index
(indicates an output item)

- Once you input the required data, the calculation is executed and the results are placed in the output column.

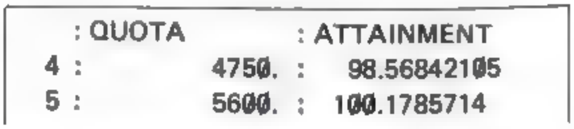
(5) Press the [ENTER] key. Now you can create the second line.



```
   : STORE NAME  : STAFF        : R
 1 : A_STORE     :          32. :
 2)              :              :
```

- If there are several output columns, use the [→] key to move the index to the last column (extreme right) and press the [ENTER] key.

(6) Enter the rest of the data.
⋮
5600 [ENTER]

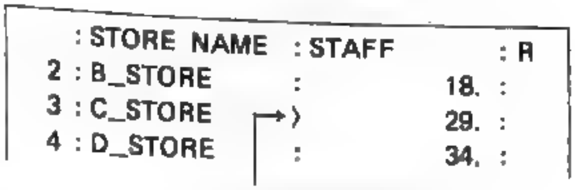


```
   : QUOTA         : ATTAINMENT
 4 :        475Ø.  :   98.568421Ø5
 5 :        56ØØ.  :  1ØØ.1785714
```

Note: Each time you input or change data, the PC-2500 performs recalculations and adjusts the display position. As a result, these calculations take time. After you press the [ENTER] key, check to see that the data has actually been input and that the index is displayed at the next column. Then enter the next data.

After you enter all the data, check to see that you have made no mistakes. You can check the data by using the ← , → , ↑ , and ↓ keys to move the index. If you notice a mistake, move the index in front of the incorrect data, enter the correct data, and press the ENTER key.

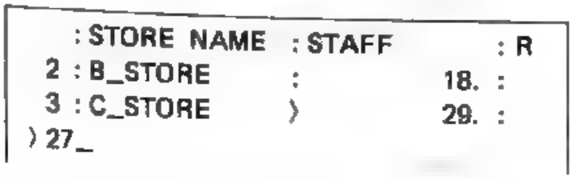
Example: Correct the staff of store C from 29 to 27.
Use the ← , → , ↑ , and ↓ keys to move the index to item "Staff" for "Store C."



```
   : STORE  NAME   : STAFF         : R
 2 : B_STORE       :          18.  :
 3 : C_STORE       )          29.  :
 4 : D_STORE       :          34.  :
```

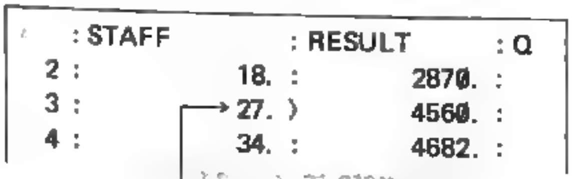
Move the index here.

27



```
   : STORE  NAME   : STAFF         : R
 2 : B_STORE       :          18.  :
 3 : C_STORE       )          29.  :
) 27_
```

ENTER



```
   : STAFF              : RESULT         : Q
 2 :               18.  :          2870.  :
 3 :               27.  )          4560.  :
 4 :               34.  :          4682.  :
```

29 has been rewritten to 27.

Be sure that the input numeric data does not exceed 23 digits (including the decimal point, sign, and E indicating the exponent part).

The data in the column in which the index is located is displayed when the ↑ key is pressed.

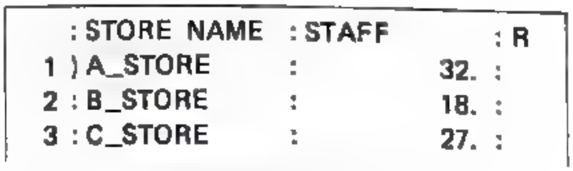
[4] Deleting or Adding a Row

There may be instances when you want to delete or add a row after creating a table. Use the methods below.

(1) Deleting a Row

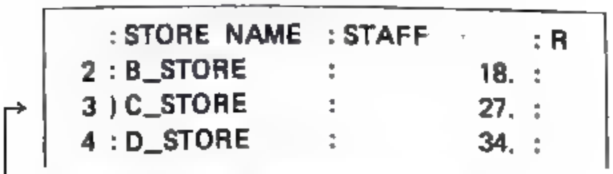
Example: Delete store C (the third row) from the table we created above.

DEF ↑



```
   : STORE  NAME   : STAFF         : R
 1 ) A_STORE       :          32.  :
 2 : B_STORE       :          18.  :
 3 : C_STORE       :          27.  :
```

↓ ↓

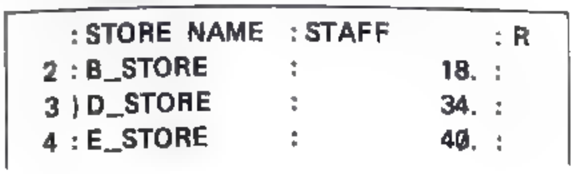


```
   : STORE  NAME   : STAFF         : R
 2 : B_STORE       :          18.  :
 3 ) C_STORE       :          27.  :
 4 : D_STORE       :          34.  :
```

Move the index to the row to be deleted.

SHIFT + DEL BS

The row for store C is cleared.



```
   : STORE  NAME   : STAFF         : R
 2 : B_STORE       :          18.  :
 3 ) D_STORE       :          34.  :
 4 : E_STORE       :          40.  :
```

(2) Adding a Row

Example: Add a row before store D (the third row).

[DEF] [WRITE ↑]

(Specifies data write)

```
   :STORE NAME  :STAFF        :R
 1) A_STORE     :          32. :
 2 : B_STORE    :          18. :
 3 : D_STORE    :          34. :
```

[↓] [↓]

(Move the index to where the line is to be added.)

```
   :STORE NAME  :STAFF        :R
 2 : B_STORE    :          18. :
 3) D_STORE     :          34. :
 4 : E_STORE
```

Move the index to the third row.

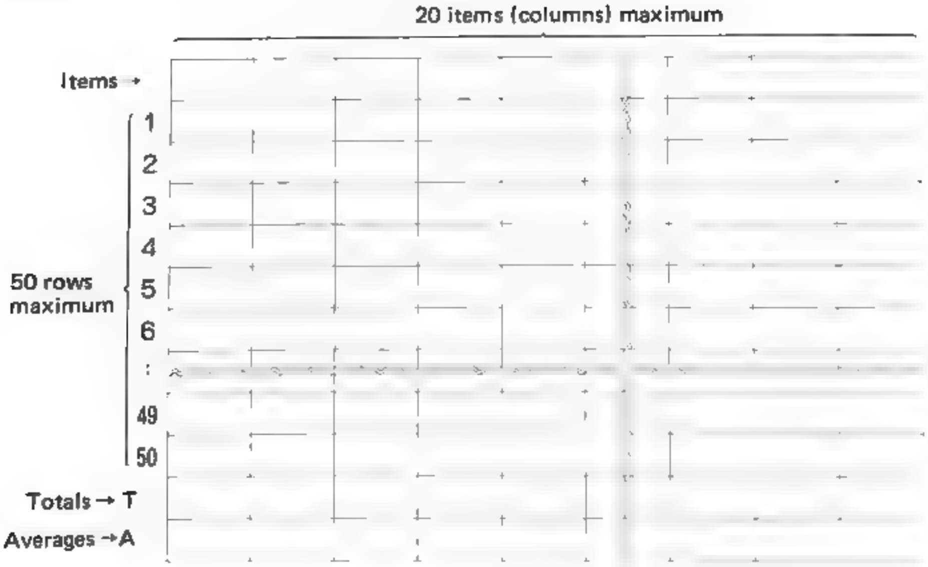
[INS]

```
   :STORE NAME  :STAFF        :R
 2 : B_STORE    :          18. :
 3)             :              :
 4 : D_STORE    :          34. :
```

A blank row is created.

Input data for store C.

[5] Table Size



As shown in the figure, a table having a maximum size of 20 items (columns) across and 50 rows (not including item, total, and average rows) can be created.
For the telephone list, the numbers for up to 254 persons can be entered.
When creating a table or telephone list of these sizes, be sure to use the optional RAM card, CE-201M or CE-202M, since the memory capacity of the PC-2500 by itself is too small.

= **RAM Card Usage and Table Size** =

The maximum size of the table which can be converted to a graph or printed using the PC-2500 depends on whether or not the RAM card is being used. Keep in mind the general sizes given below when creating your table. (The sizes given are for either a table or a telephone list.)

PC-2500 without card	
Table	10 items x 7 rows
Telephone Book	For 80 persons

Type of RAM card	Card used as expansion memory		Card used while removed from the PC-2500
CE-201M (8K bytes)	Table	20 items x 27 rows	20 items x 18 rows
	Telephone book	For 254 persons	For 254 persons
CE-202M (16K bytes)	Table	20 items x 50 rows	20 items x 42 rows
	Telephone book	For 254 persons	For 254 persons

- The values given above vary if the vertical/horizontal ratio of your table is different or if you store several tables.
- The size of the table you can create decreases if:
  - Memory is occupied by a BASIC program, array variables, etc.
  - Data (e.g. telephone numbers) was entered for the telephone list.
  - Several tables are already stored.

Note: Data for 254 persons can be entered into the telephone list. However, the more data items there are, the longer it will take to sort (see page 326) or to search (see page 329).
Data for about 50 persons is practical.
Enter the data in alphabetical order if possible.
If you add data later, insert the data at the appropriate location to maintain the alphabetical order. (See page 318.)

3-3) Table Selection

When printing a formula or table, creating a graph, or checking and correcting data, you must specify a table.
Confirm or change the table you selected by the method given below.

(1) Press the DEF ↓ keys. The titles of the stored tables will be displayed. The title of the currently specified table is indicated by the index.

```
  * TELEPHONE BOOK *   PRODUCT
 〉STORE NAME
```

↑ Index (Indicates the specified table)

(2) To change the specified table, move the index to the desired table by using the ← , → , ↑ , and ↓ keys, then press the ENTER key.

- Once you specify (select) a table, it is retained until you create a new table, delete the specified table, or change the specified table.
When you create a new table, it becomes the currently specified table.

3-4) Setting the Display Format

〈 Procedure 〉

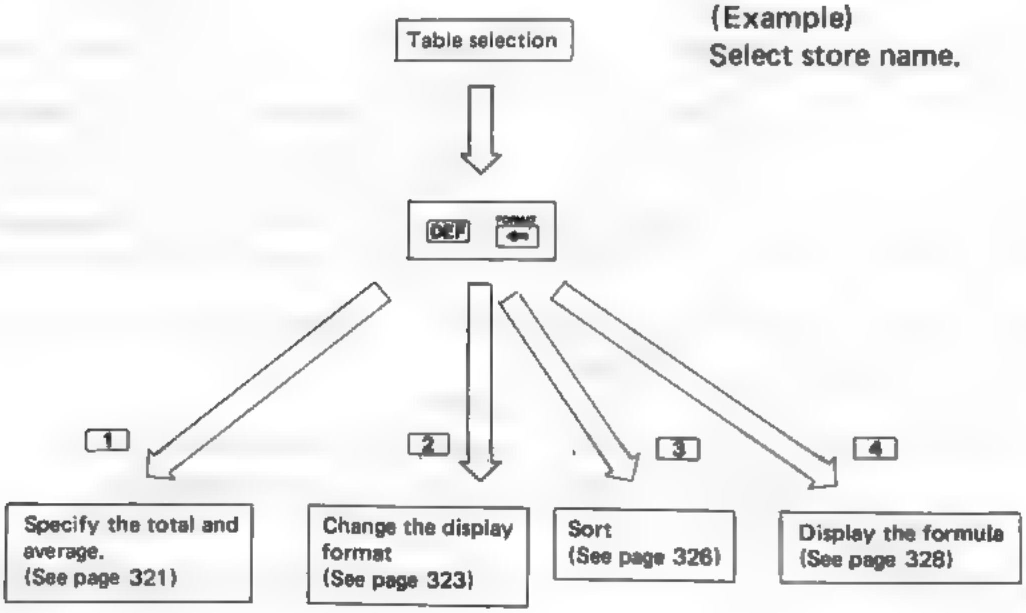
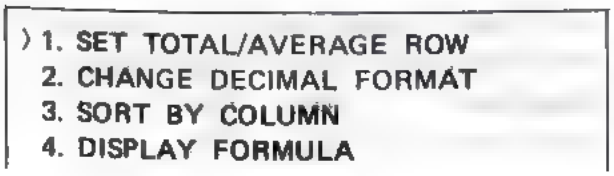


Table selection
DEF FORMAT ←
1
2
3
4
Specify the total and average. (See page 321)
Change the display format (See page 323)
Sort (See page 326)
Display the formula (See page 328)

(Example)
Select store name.

By pressing DEF FORMAT ← , the figure shown on the right will be displayed. You can select the desired function from this display.



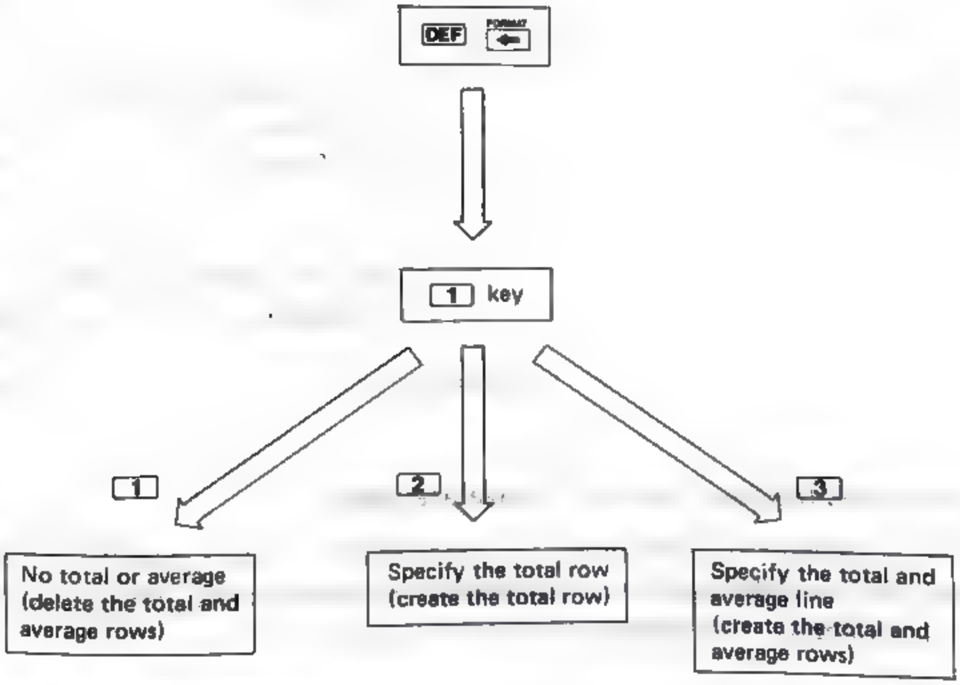
```
〉1. SET TOTAL/AVERAGE ROW
  2. CHANGE DECIMAL FORMAT
  3. SORT BY COLUMN
  4. DISPLAY FORMULA
```

(Description)

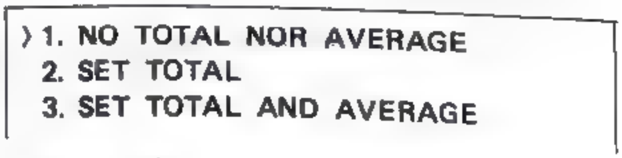
1. Key 1 to specify the totals and averages
2. Key 2 to change the display format
3. Key 3 to sort
4. Key 4 to display the formula

[1] Specifying the Total and Average

〈Procedure〉


DEF FORMAT ←
1 key
1
2
3
No total or average (delete the total and average rows)
Specify the total row (create the total row)
Specify the total and average line (create the total and average rows)

After obtaining the display shown above, press the 1 key. The display changes to the one shown on the right.

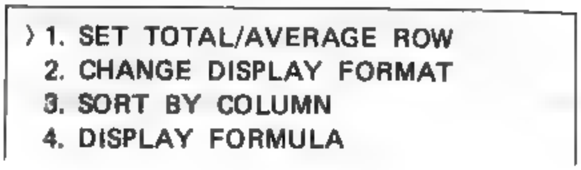


```
〉1. NO TOTAL NOR AVERAGE
  2. SET TOTAL
  3. SET TOTAL AND AVERAGE
```

(Description)

1. Key 1 if total and average are not needed
2. Key 2 to create the total row
3. Key 3 to create the total and average rows

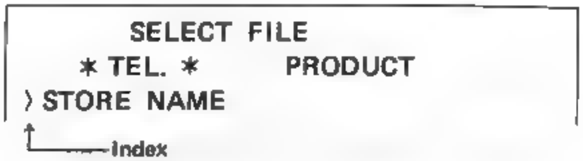
Let's add a total row and an average row to the table titled "Store Name." Press the 3 key. The total and average rows will be specified.



```
〉1. SET TOTAL/AVERAGE ROW
 2. CHANGE DISPLAY FORMAT
 3. SORT BY COLUMN
 4. DISPLAY FORMULA
```

Now let's check it. Press the keys as shown below.

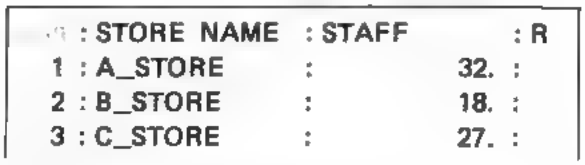
DEF SELECT ↓



```
       SELECT FILE
   * TEL. *      PRODUCT
〉STORE NAME
```

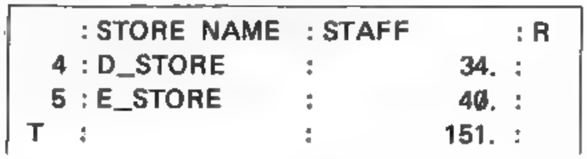
Index

ENTER



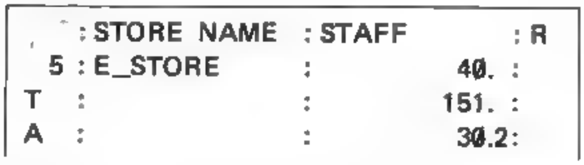
```
   : STORE NAME  : STAFF      : R
 1 : A_STORE     :       32.  :
 2 : B_STORE     :       18.  :
 3 : C_STORE     :       27.  :
```

SHIFT + ↓



```
   : STORE NAME  : STAFF      : R
 4 : D_STORE     :       34.  :
 5 : E_STORE     :       4Ø.  :
T  :             :      151.  :
```

↓



```
   : STORE NAME  : STAFF      : R
 5 : E_STORE     :       4Ø.  :
T  :             :      151.  :
A  :             :       3Ø.2:
```

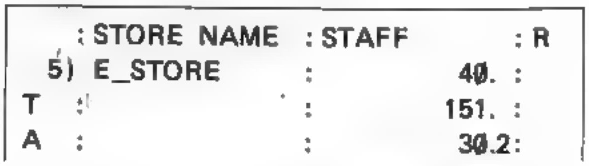
Doing this creates the total line and the average rows.

Note: The achievement is obtained in this example. If you specify the total and average rows, the total and average for the achievement will also be obtained. Unfortunately, these values have no significance.

Since the character item is blank, enter names to represent the total and average (e.g. Total, Average). (The displayed T and A will not be printed.)

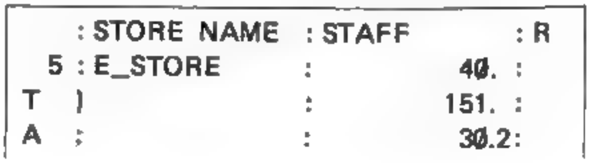
(Example)

DEF WRITE ↑ Calls the table or index.



```
   : STORE NAME  : STAFF      : R
 5〉E_STORE      :       4Ø.  :
T  :             :      151.  :
A  :             :       3Ø.2:
```

↓



```
   : STORE NAME  : STAFF      : R
 5 : E_STORE     :       4Ø.  :
T  〉            :      151.  :
A  :             :       3Ø.2:
```

TOTAL

```
    : STORE NAME   : STAFF        : R
  5 : E_STORE      :          4Ø. :
T   )              :         151. :
) TOTAL_
```

ENTER

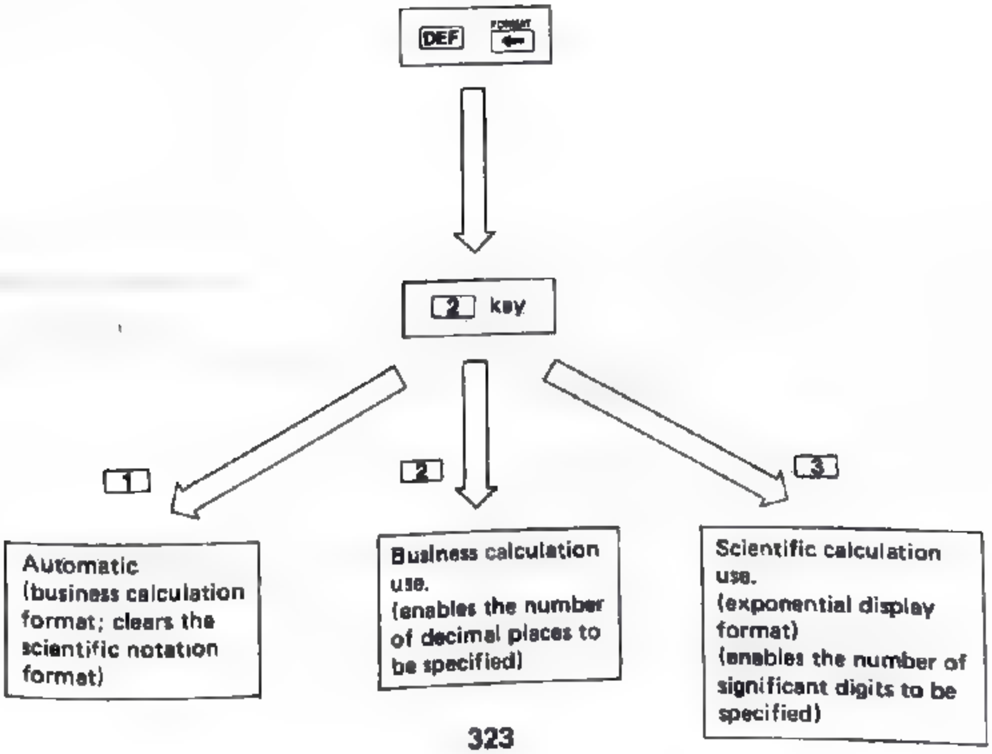
```
    : STORA NAME   : STAFF        : R
  5 : E_STORE      :          4Ø. :
T   : TOTAL        :         151. :
A   :              :          3Ø.2:
```

Move the index to the total row and the average row to enter their names as shown.

> **Function of SHIFT + ↓ and SHIFT + ↑**
>
> When a table is displayed, pressing the ↓ key moves the index to the next row (one row down) and pressing the ↑ key moves the index to the previous row (one row up).
> Similarly, pressing the SHIFT + ↓ keys moves the index 3 rows down and pressing the SHIFT + ↑ keys moves the index 3 rows up. You can use these keys to call 3 rows of data to the display to check the data.

[2] Changing the Display Format

(Procedure)


DEF FORMAT ←
2 key
1
2
3
Automatic (business calculation format; clears the scientific notation format)
Business calculation use. (enables the number of decimal places to be specified)
Scientific calculation use. (exponential display format) (enables the number of significant digits to be specified)

Change the display to the one shown on page 321 and press the 2 key. The display changes to the one shown on the right.

```
> 1. GENERAL FORMAT
  2. BUSINESS FORMAT
  3. SCIENTIFIC FORMAT
```

(Description)
Press

1. The 1 key to display the results like a regular calculator without specifying the number of digits.
2. The 2 key to display the results for business calculation use (fixed number of decimal places).
3. The 3 key to display the results for scientific calculation use (exponential display).

(1) Setting for Business Calculations

While in the above display, press the 2 key.

```
 DECIMAL PLACES: Ø–9
> 2
```

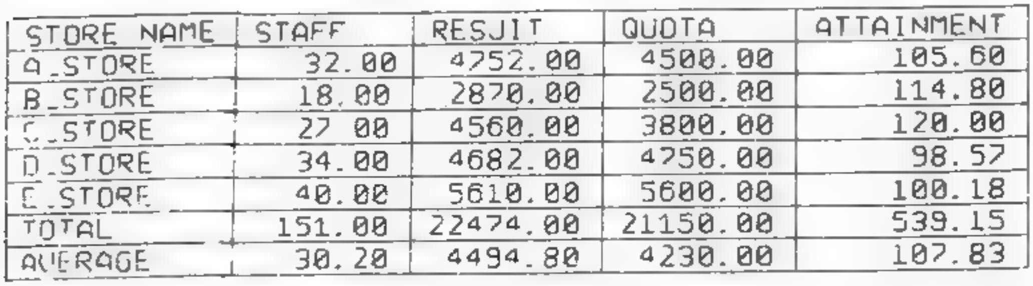
Blinking cursor

The number of decimal places is initially shown to be 2. Press the ENTER key to set the display to 2 decimal places.
You can set the display from 0 to 9 decimal places by pressing the corresponding number key and then the ENTER key.

```
12.3Ø
```

Two decimal places, the second decimal place is rounded. (The number of decimal places can be set from 0 to 9.)

(**Print Example**) The number of decimal places set at 2.



STORE NAME	STAFF	RESJIT	QUOTA	ATTAINMENT
A_STORE	32.00	4752.00	4500.00	105.60
B_STORE	18.00	2870.00	2500.00	114.80
C_STORE	27 00	4560.00	3800.00	120.00
D_STORE	34.00	4682.00	4750.00	98.57
E_STORE	40.00	5610.00	5600.00	100.18
TOTAL	151.00	22474.00	21150.00	539.15
AVERAGE	30.20	4494.80	4230.00	107.83

For these values, the second decimal place is rounded. Calculations for the output column (horizontal calculations) and for the total and average rows are based on rounded data. Therefore, when using the PC-2500 for business calculations, specify the display format before inputting data.

(2) Setting for Scientific Calculations

While in the display shown on page 323, press the 3 key. The display will change to the one shown on the right. The number of significant digits is initially shown to be 6.

```
 SIGNIFICANT FIGURES: 1–1Ø
> 6
```

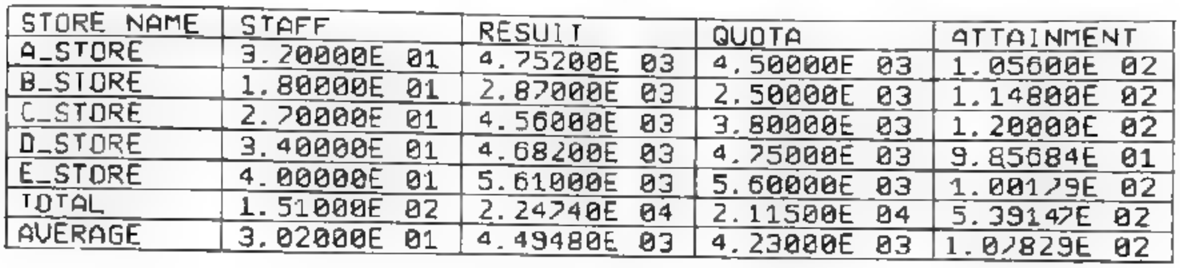
Blinking cursor

Press the [ENTER] key to set the display to 6 significant digits. You can set the display from 1 to 10 significant digits by pressing the corresponding number key(s) and then the [ENTER] key.

$\underbrace{1.23456}_{}$E Ø5

6 significant digits (The number of significant digits can be set from 1 to 10.)

(Print Example) The number of significical figures set at 6.



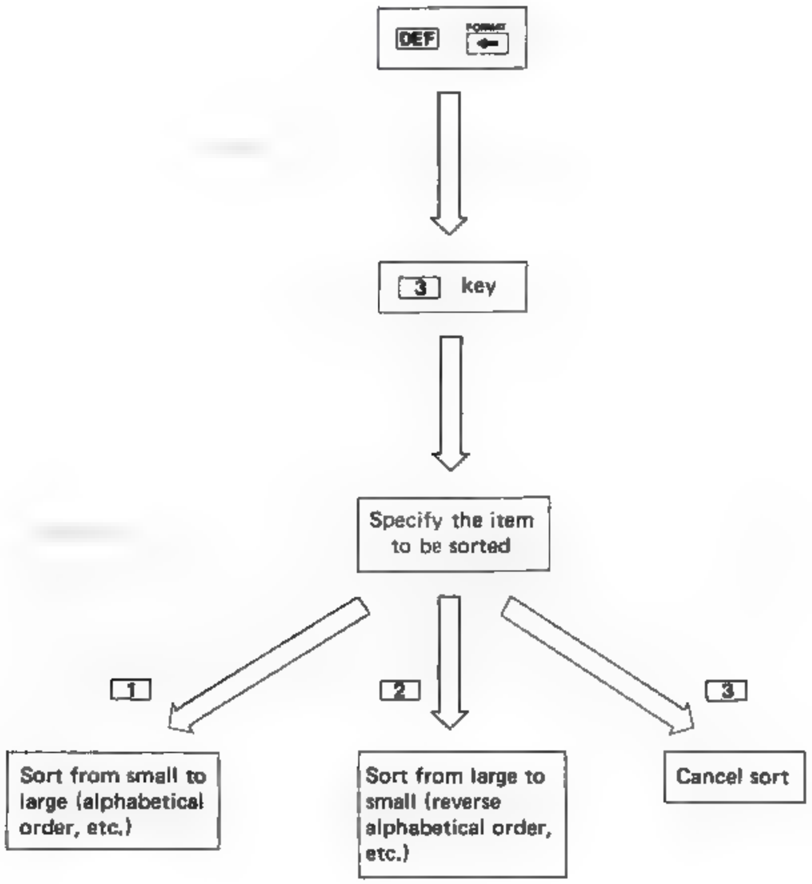
STORE NAME	STAFF	RESUIT	QUOTA	ATTAINMENT
A_STORE	3.20000E 01	4.75200E 03	4.50000E 03	1.05600E 02
B_STORE	1.80000E 01	2.87000E 03	2.50000E 03	1.14800E 02
C_STORE	2.70000E 01	4.56000E 03	3.80000E 03	1.20000E 02
D_STORE	3.40000E 01	4.68200E 03	4.75000E 03	9.85684E 01
E_STORE	4.00000E 01	5.61000E 03	5.60000E 03	1.00179E 02
TOTAL	1.51000E 02	2.24740E 04	2.11500E 04	5.39147E 02
AVERAGE	3.02000E 01	4.49480E 03	4.23000E 03	1.07829E 02

Note: This table is printed in a display format for scientific calculation for comparative purposes.

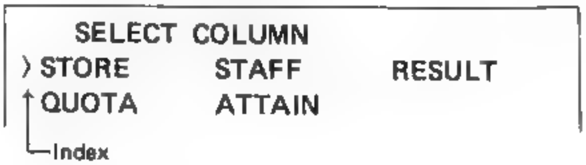
Specifying the number of significant digits affects only the display and not the data stored within the PC-2500. However, calculations for the output column (horizontal calculations) and for the total and average rows are based on rounded data.

[3] Sorting

〈Procedure〉


DEF
FORMAT
←
3 key
Specify the item to be sorted
1
2
3
Sort from small to large (alphabetical order, etc.)
Sort from large to small (reverse alphabetical order, etc.)
Cancel sort

Change the display to the one shown on page 321 and press the 3 key. The display changes to the one shown on the right.



```
   SELECT COLUMN
〉STORE       STAFF       RESULT
 QUOTA       ATTAIN
```

Specify the item on which the sort operation is to be based.

Move the index to the position of the item to be sorted.

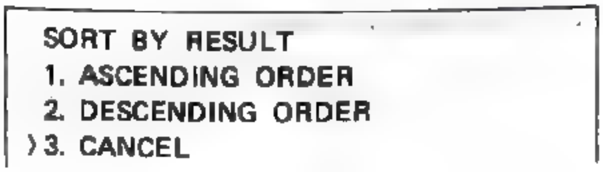
Let's sort "staff" from large to small numbers.

Note: The sort operation does not rearrange only the data for the specified item. It rearranges the rows in the table so that the data for the specified item is in ascending order or descending order.

Note: The data for a character item is sorted according to its character code (see page 221). Consequently, the data may not be sorted in correct alphabetical order (e.g. as in a dictionary) if the data consists of a mixture of upper case and lower case letters.

⇨ (Move the index)

ENTER



SORT BY RESULT
1. ASCENDING ORDER
2. DESCENDING ORDER
›3. CANCEL

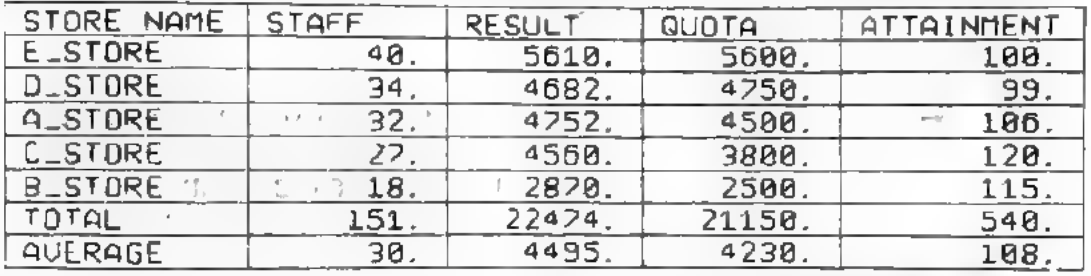
(Description)
Press
1 The 1 key to sort from small to large.
2. The 2 key to sort from large to small.
3. The 3 key to cancel the sort operation.

Then press the 2 key. This sorts the data.

** SORTING **

(Print Example)

Sorted based on staff from large to small (total and average rows are omitted here.)



STORE NAME	STAFF	RESULT	QUOTA	ATTAINMENT
E_STORE	40.	5610.	5600.	100.
D_STORE	34.	4682.	4750.	99.
A_STORE	32.	4752.	4500.	106.
C_STORE	27.	4560.	3800.	120.
B_STORE	18.	2870.	2500.	115.
TOTAL	151.	22474.	21150.	540.
AVERAGE	30.	4495.	4230.	108.

This table is rearranged so that the number of staff is in order from large to small.

- Many band graphs and pie charts show data in ascending order. The sort function is convenient when preparing data for these graphs or charts.

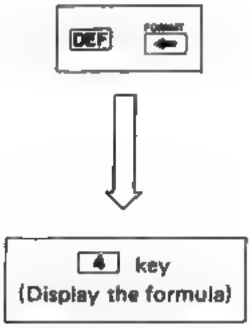
Note: Once you sort the data, the data usually cannot be restored to its original order. (In the above example, sorting the store names in alphabetical order restores the original order.) Therefore, when you input the formula and create a table, it is convenient to have an item to list the serial numbers.

"PRODUCT": No.: A_STORE: ....

PRODUCT	No.	A_STORE
TV	1.	1240.
VIDEO	2.	980.

[4] Checking the formula

〈Procedure〉



(1) DEF ←

```
〉1. SET TOTAL / AVERAGE ROW
 2. CHANGE DECIMAL FORMAT
 3. SORT BY COLUMN
 4. DISPLAY FORMULA
```

(2) Press 4 key.

(3) Correct data using the DEL, INS, ↑, ↓, →, and ← keys. Press ENTER to end correction.

```
   ** DISPLAY FORMULA **
〉"STORE NAME" : STAFF : ATTA
INMENT = RESULT / QUOTA * 100
```

(4) When only columns are corrected, the display changes as shown to the left.

```
   ** EDIT FORMULA **
 1. EDIT
〉2. CANCEL
```

Press 1 to create a new table.

When the number of columns is changed, the display changes as shown to the left.

```
   ** EDIT FORMULA **
 1. EDIT OLD DATA
〉2. CANCEL
```

Press 1 to create a new table.

When the number of columns is changed, the contents of the old table are last.

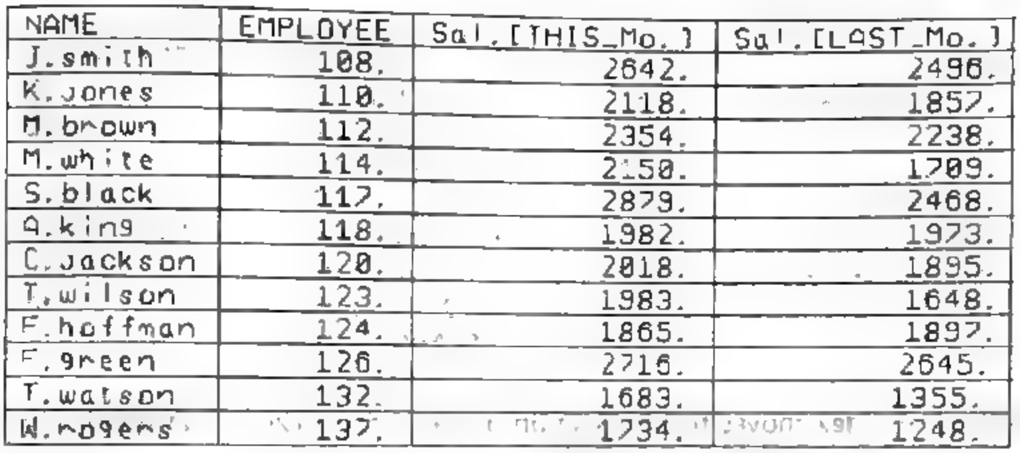
(5) The new table is created.

```
   ** COMPUTING **
```

3-5) Searching

It is a tedious task to use the ↓ and ↑ keys to search for a specific item of data if you have a large amount of data in each column. The PC-2500 is equipped with a function to search for and display the data you specify. This function is called the search function.

Suppose that you have the table shown below in your PC-2500. We will describe the search function using this table.



NAME	EMPLOYEE	Sal.[THIS_Mo.]	Sal.[LAST_Mo.]
J.smith	108.	2642.	2496.
K.jones	110.	2118.	1857.
M.brown	112.	2354.	2238.
M.white	114.	2150.	1709.
S.black	117.	2879.	2468.
A.king	118.	1982.	1973.
C.jackson	120.	2018.	1895.
T.wilson	123.	1983.	1648.
F.hoffman	124.	1865.	1897.
F.green	126.	2716.	2645.
T.watson	132.	1683.	1355.
W.rogers	137.	1734.	1248.

(**Procedure**)

DEF ↑ — Start the BUSINESS software and select the desired table
The index (✱) is displayed at the first row of the first column.

Specify the item (column) — Move the index by pressing the → key to the item (column) containing the data you want to retrieve.

MODE — Set to the search mode.

Input data — Input the data that you want to retrieve.

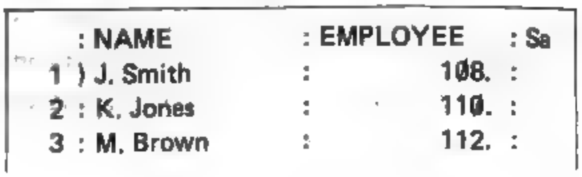
ENTER

The position of the input data is display. — The index moves to the position of the specified data in the table.

(Example 1) Check the name and sales for the person with name code 124.

(1) DEF ↑

Call the table.
(Move the index to the first row of first column.)

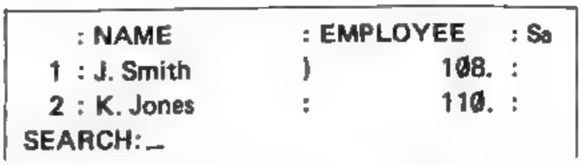


```
    : NAME            : EMPLOYEE    : Sa
  1 ) J. Smith        :        1Ø8. :
  2 : K. Jones        :        11Ø. :
  3 : M. Brown        :        112. :
```

(2) → Move the index to the name code item.

(3) MODE

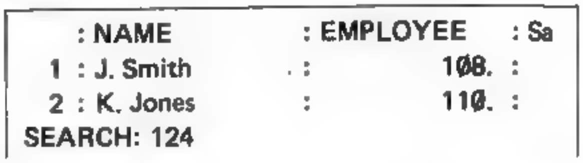
Set to the search mode.



```
    : NAME            : EMPLOYEE    : Sa
  1 : J. Smith        )        1Ø8. :
  2 : K. Jones        :        11Ø. :
SEARCH:_
```

(4) 124

Enter the data. (name code)



```
    : NAME            : EMPLOYEE    : Sa
  1 : J. Smith        :        1Ø8. :
  2 : K. Jones        :        11Ø. :
SEARCH: 124
```

ENTER The index moves to the position of name code 124. That row is displayed. Check that the name is Hoffman.

- Move the index and check the sales.

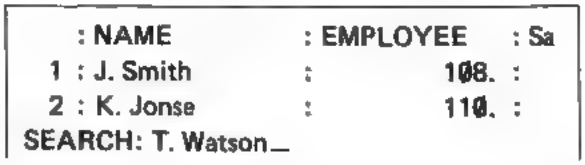
(Example 2) Check the sales for Watson.

(1) DEF ↑

Move the index to the first row of the name item.

(2) MODE Set to the search mode.

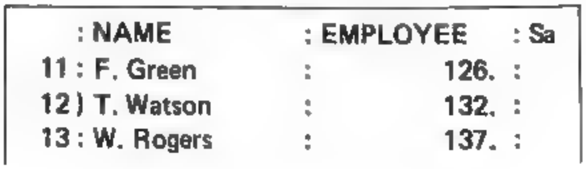
(3) T. Watson



```
    : NAME            : EMPLOYEE    : Sa
  1 : J. Smith        :        1Ø8. :
  2 : K. Jonse        :        11Ø. :
SEARCH: T. Watson_
```

(4) ENTER

The row containing Watson (11th line) is retrieved.



```
    : NAME            : EMPLOYEE    : Sa
 11 : F. Green        :        126. :
 12 ) T. Watson       :        132. :
 13 : W. Rogers       :        137. :
```

(5) → . . . Check the sales.

- If the data to be searched consists of characters, it can be abbreviated. For example, T. W, T. Wa or T. Wat can be entered to search for T. Watson. Note, however, that if T. W was entered in this example, T. Wilson will be retrieved first.

〈Search Conditions〉

(1) The search operation is performed only on data under the specified item.

(2) The search operation is performed on rows below the one where the index is during the data write mode.
When the table is retrieved after being selected (when the index is not displayed), the index is assumed to be at the top row being displayed. The entire column is searched starting at the index.

(3) The search operation for a character item proceeds by comparing, one character at a time, the input characters and the characters in each row.
If all the input characters match the contents in a row, that row is retrieved.

(4) If the input value and the data matches for a numeric item, the row containing the data is retrieved.

(5) The search operation proceeds by comparing, one row at a time, from the row where the index is currently positioned or from the top row being displayed.

(Example)

```
: NAME       :
: J. Smith   :
: K. Jones   :
: M. Brown   :
: W. White   :
: S. Black   :
: A. King    :
: C. Jacson  :
: T. Wilson  :
: F. Hoffman :
: T. Watson  :
: W. Rogers  :
```

When "W" is used to search for data, W. White is retrieved when the index is at the 1st row (1st to 3rd rows) and W. Rogers is retrieved when the index is at the 4th row (4th and 11th rows). Therefore, all the names starting with "W" under the item can be checked by repeating MODE W ENTER MODE W ENTER . . . after moving the index to the top row. If data matching the input data is not found, the index does not move.

3-6) Printing

(Procedure)

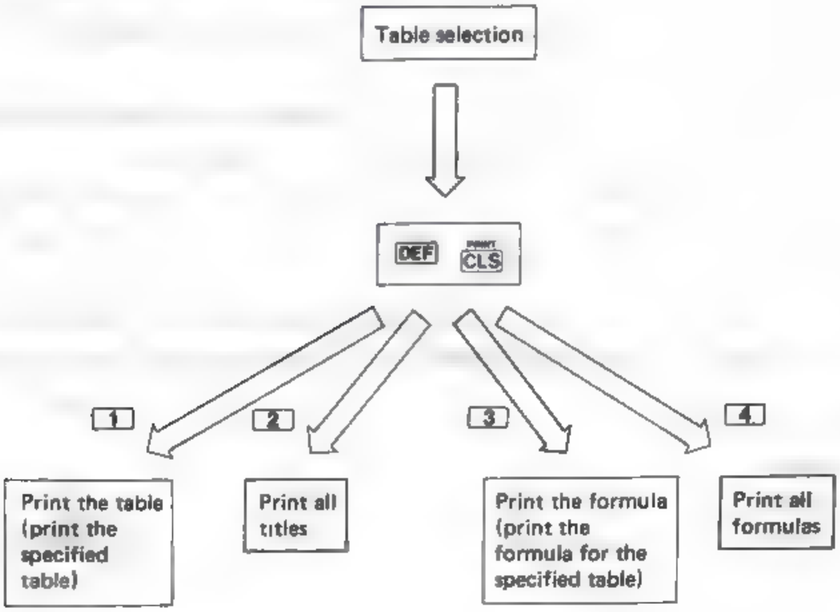
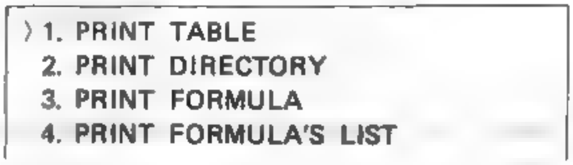


Table selection
DEF
PRINT
CLS
1
2
3
4
Print the table (print the specified table)
Print all titles
Print the formula (print the formula for the specified table)
Print all formulas

Press DEF CLS to display the figure shown on the right. (Print selection display)



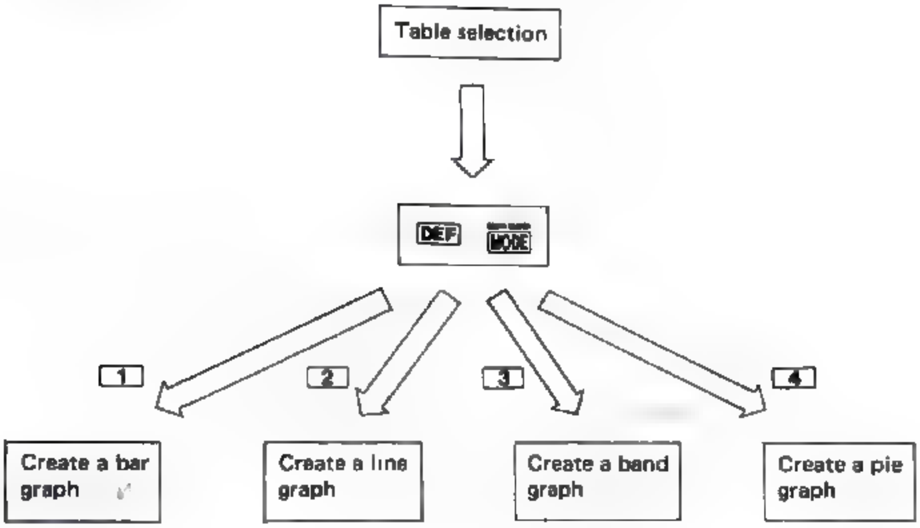
```
〉1. PRINT TABLE
 2. PRINT DIRECTORY
 3. PRINT FORMULA
 4. PRINT FORMULA'S LIST
```

Press the 1 , 2 , 3 , or 4 key to print the table, all titles, formula, or all formulas, respectively.

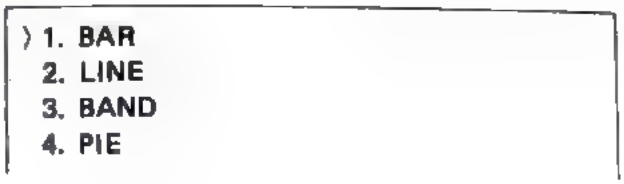
- If the number of rows in the table (includes those for items, total, and average) exceeds 21, the printing will be divided into 2 since the width of the paper does not permit them to be printed all at once.

3-7) Printing Graphs

〈 Procedure 〉



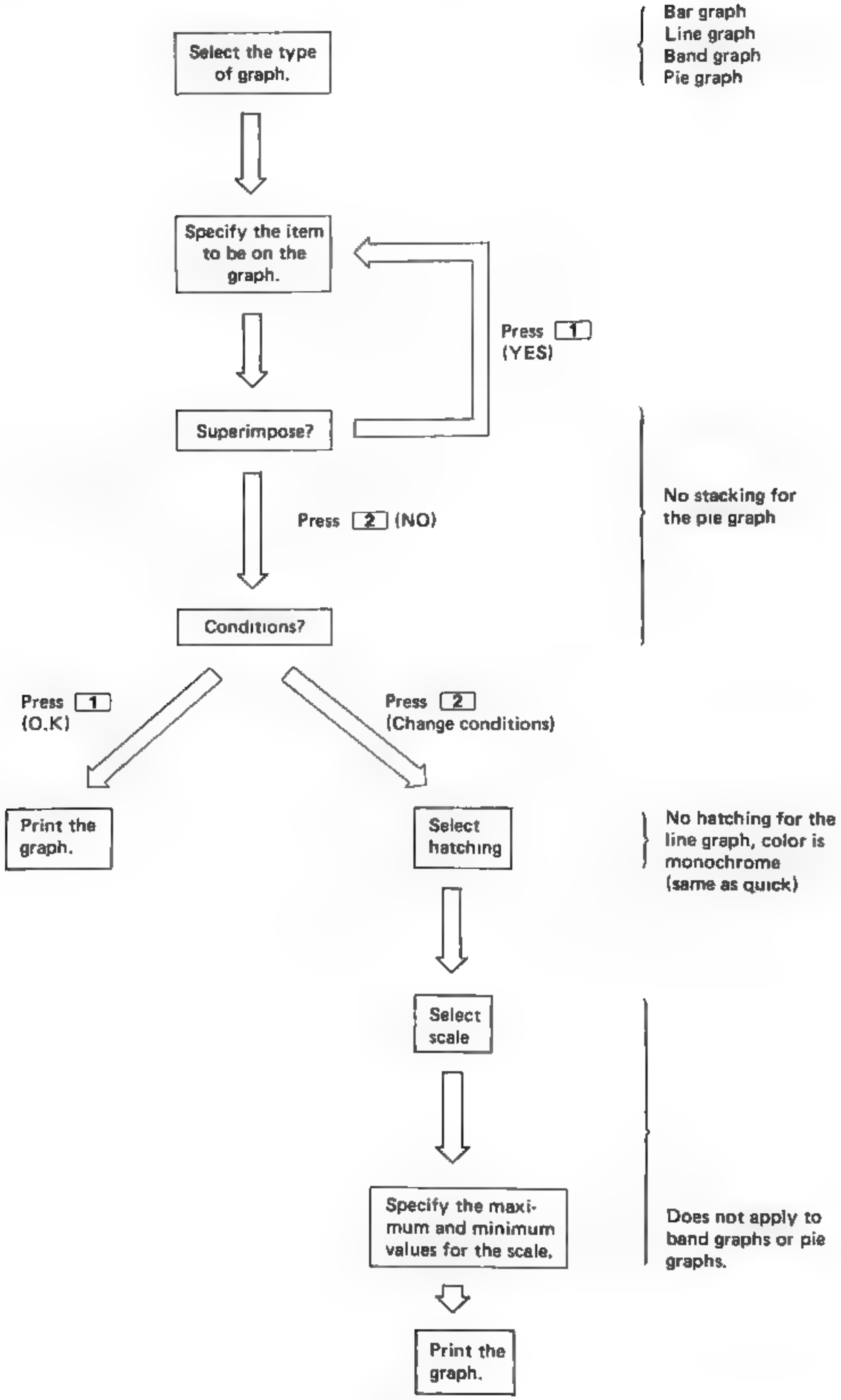
Press the DEF MODE keys to display the figure shown on the right. From this screen (graph selection screen) you can create various types of graphs.



```
〉1. BAR
  2. LINE
  3. BAND
  4. PIE
```

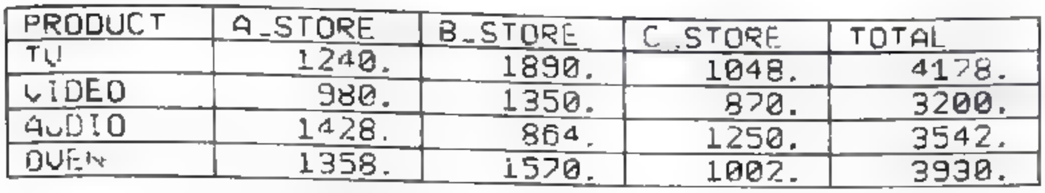
[1] Printing Graphs

⟨ Procedure ⟩


Select the type of graph.
Bar graph
Line graph
Band graph
Pie graph
Specify the item to be on the graph.
Press 1 (YES)
Superimpose?
Press 2 (NO)
No stacking for the pie graph
Conditions?
Press 1 (O.K)
Press 2 (Change conditions)
Print the graph.
Select hatching
No hatching for the line graph, color is monochrome (same as quick)
Select scale
Specify the maximum and minimum values for the scale.
Does not apply to band graphs or pie graphs.
Print the graph.

Generally, you can print a graph by following the procedures shown on the previous page. As an example, we will describe the procedures to print a bar graph.

(Example): Create a bar graph from the following table.



PRODUCT	A_STORE	B_STORE	C_STORE	TOTAL
TV	1240.	1890.	1048.	4178.
VIDEO	980.	1350.	870.	3200.
AUDIO	1428.	864.	1250.	3542.
OVEN	1358.	1570.	1002.	3930.

- Stack stores A, B, and C.
- Draw the graph in the quick mode.
- Draw scale lines.
- Maximum value (MAX) of the scale is A (auto).
  Minimum value (MIN) is 500.

(1) Select the table. (DEF ⬇)

(2) DEF MODE [illegible]

```
)1. BAR
 2. LINE
 3. BAND
 4. PIE
```

(3) 1 (select the BAR graph)

```
   ANY OTHER COLUMNS?
)PRODUCT    A_STORE    B_STORE
 C_STORE    TOTAL
```

(4) Specify the item to be on the graph by pressing ✱ . For example, move the index to store A and press ✱ , then move the index to store B and press ✱ . → , ↓ , ← , ↑ character items cannot be specified for graphs. Character items are usually printed as titles. By specifying a colon (:) preceding an item, that item can be printed as a title.

(5) ENTER

```
 ANY OTHER COLUMNS?
 1. YES
)2. NO
```

Once you specify the item, the program asks if you want to stack data on the graph.

Note: To stack means to show data for 2 or more items on the same graph.

(6) Press the 1 key to stack data on the graph. (If you do not want to stack data, go to step 8.)

```
   ANY OTHER COLUMNS?
 PRODUCT    *A_STORE   )B_STORE
 C_STORE    TOTAL
```

* Indicates an item already specified to be on the graph.

(7) Repeat steps 4 through 6 and specify the items to be stacked.

Note: If you specify the wrong item, move the index to that item and press the ENTER key to clear it. (The * mark will disappear.)

(8) After specifying the items, press the 2 key (or ENTER key) when the figure shown on the right is displayed.

```
ANY OTHER COLUMNS?
1. YES
2. NO
```

The printing conditions will be displayed.

```
FRAME    MAX = A
NO GRID  MIN = A
) 1. DRAW GRAPH
  2. SET GRAPH FORMAT
```

(Conditions)

Quick: Quickly print only the outline.

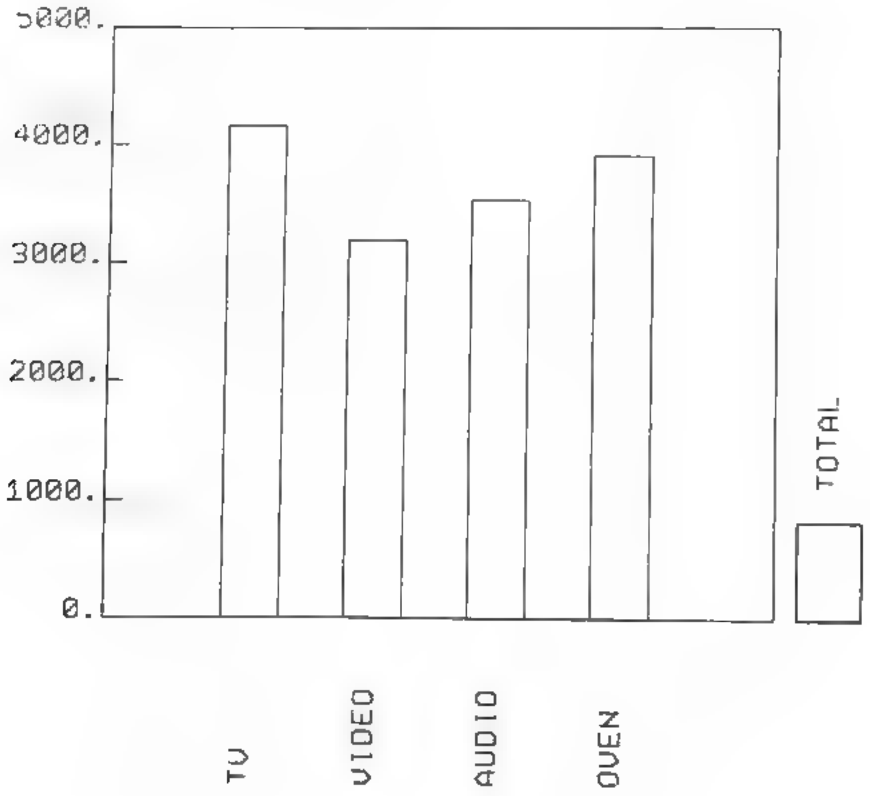
No scale: Do not print scale lines.

MAX = A: Maximum value of the scale is automatically set.

MIN = A: Minimum value of the scale is automatically set.

- To print with these conditions, press the 1 key (ENTER key).

Sample Printout using the Conditions Above (only data for the totals item is printed)



(9) Press the 2 key to change a condition. The display will change to the one shown on the right.

```
〉1. FRAME (DRAW QUICKLY)
 2. HATCH (W/BLACK HATCH)
 3. COLOR (W/COLOR HATCH)
```

Hatching: Shade the area with lines.

- If the 0 key is pressed, the legend at the side of the graph will be omitted. Then, when the 1, 2, or 3 key is pressed.
  ∗ No legend printed ∗
  will be displayed.

(10) Specify the color. Press the 3 key. The program asks you if you want the scale lines.

```
 DRAW WITH GRID?
〉1. NO GRID (SCALE ONLY)
 2. SCALE AND GRID
```

(11) Draw the scale lines. Press the 2 key.

```
MAX = A
```

(12) Enter the maximum value of the scale.
Enter A here (for auto).
ENTER

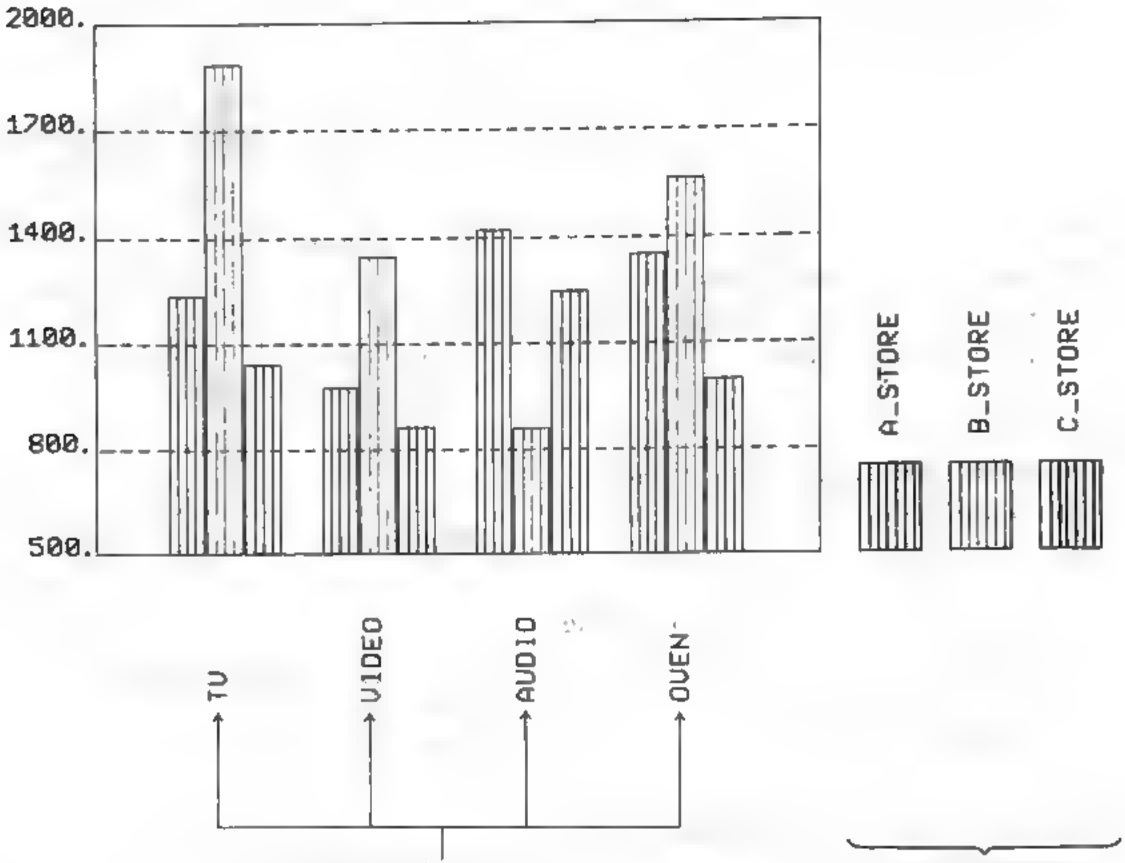
```
MAX = A
MIN = A
```

(13) Enter the minimum value of the scale.
Enter 500 here.
Press the CLS key, then
500 ENTER

```
   ** DRAWING **
TITLE: PRODUCT
```

- The graph will be printed.

〈 **Print sample** 〉



The character size becomes smaller for words having more than 8 characters.

This part is not printed if the $\boxed{0}$ key is pressed in step 9.

- The same graph can be printed as many times as desired.

= **The Scale** =

A scale is added to bar and line graphs in the following way.

(1) The scale is automatically added and starts from the specified minimum value and ends beyond the specified maximum value.

(2) The scale is spaced automatically and divided into 4 or 5 divisions.

(3) The maximum value (MAX) and minimum value (MIN) can be specified within the following ranges:

$Ø < \text{MAX} < 8.Ø\text{E}99$

$Ø \leq \text{MIN} < 8.Ø\text{E}99$

The input value for each must not exceed 10 columns (including the decimal point and the E representing the exponent part). If a value exceeding 10 columns is input, the excess part is ignored.

(Example) .12345.6789012

Only these 10 columns are significant.

1.234567E89

Only the first 10 columns are significant.

Note: Be sure that the E for the exponent part is in upper case. If a lower case is used, it and the columns following it will be ignored.

(4) The maximum value will be automatically set by entering A for the maximum value (to specify auto).

(5) The minimum value is set to 0 and the maximum value is set automatically according to the value of the data in the following cases:

a. When the value of the maximum value and minimum value contadict (i.e. the minimum value is greater than the maximum value).

b. When the specified minimum value and maximum value are negative.

c. When A (for auto) has been set for the maximum value and the specified minimum value is greater than the largest value of data.

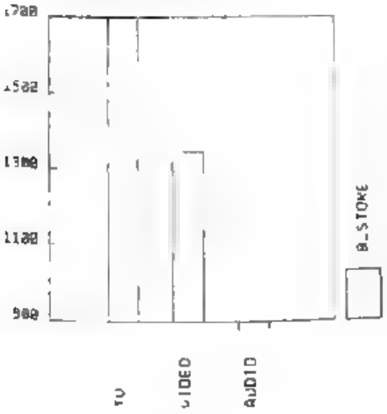
(6) When the maximum value has been set to A and the data values are all 0s, the maximum value is assumed to have been set to 1, which is determined as the maximum value of the scale.

(7) If a data value is outside the range, the specified maximum value and minimum value of the scale, it is printed in the following way:

(8) Negative data can be printed as bar graphs or line graphs.

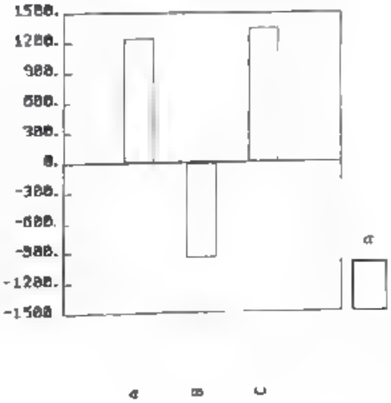
⟨ Print sample ⟩

Indicates that the data is greater than 1700.

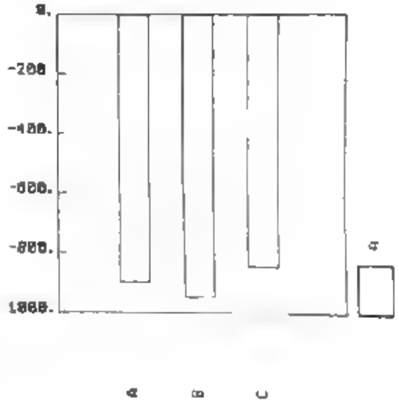


Indicates that the data is less than 900.

When data contains positive and negative value.



When data contains negative value only.



Note: If one of the following is displayed when DEF MODE are pressed, is mode before DEF MODE was pressed may be restored.

** MEMORY OVERFLOW **

** INVALID DATA **

These indicate that there is insufficient memory to create a graph.

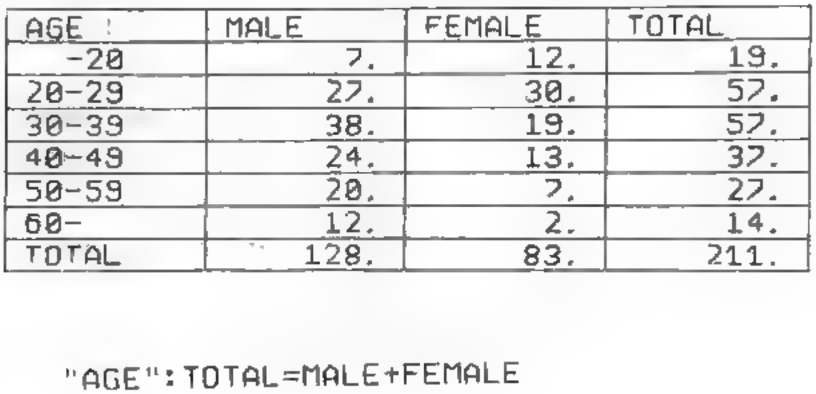
[2] Features of the Different Graphs

(1) Bar Graph

The bar graph is suited for showing the quantity of each item and is convenient if used to compare the quantity or change in quantity of 2 or more items.

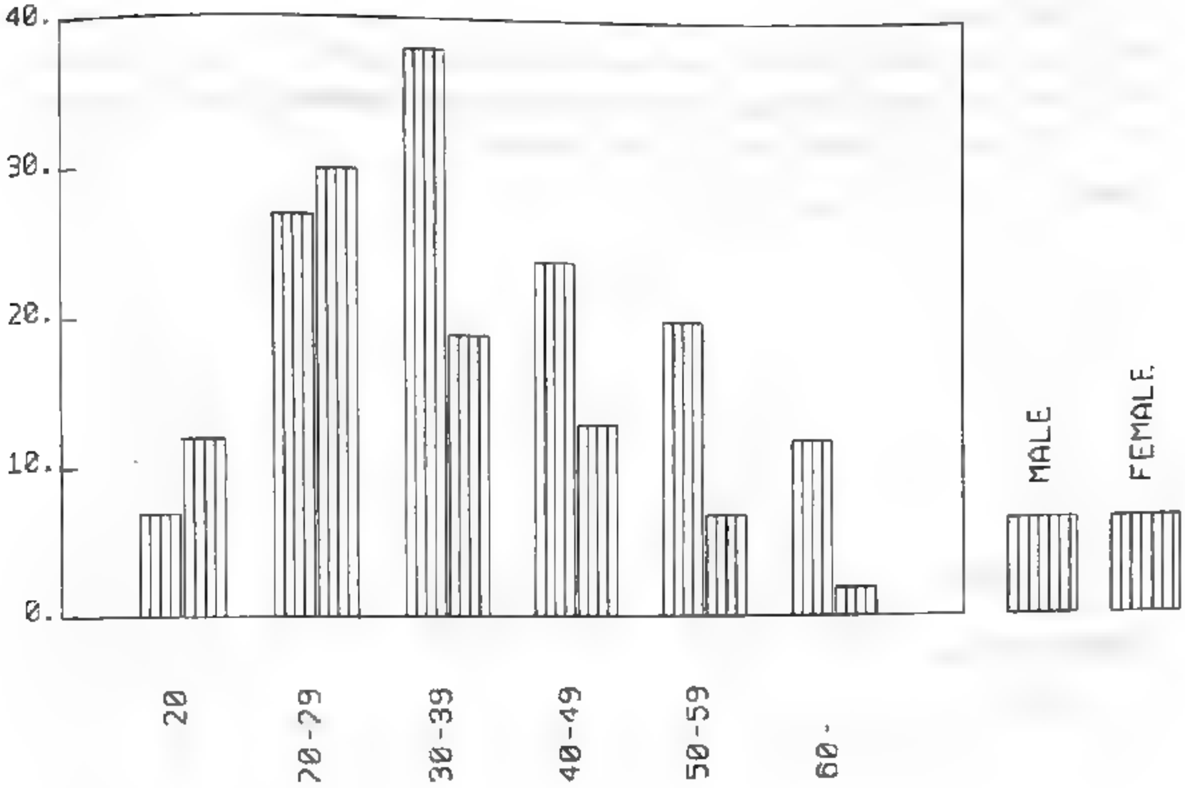
The bar graph can be easily used by the sales, personnel, or management division to graph the sales of each sales office.

Example: Graph showing the staff divided by sex and age



AGE	MALE	FEMALE	TOTAL
-20	7.	12.	19.
20-29	27.	30.	57.
30-39	38.	19.	57.
40-49	24.	13.	37.
50-59	20.	7.	27.
60-	12.	2.	14.
TOTAL	128.	83.	211.

```
"AGE":TOTAL=MALE+FEMALE
```



(Print Conditions)

- Stack (male, female)
- Hatching
- No scale lines
- MAX = A
- MIN = Ø

= **Functions** =

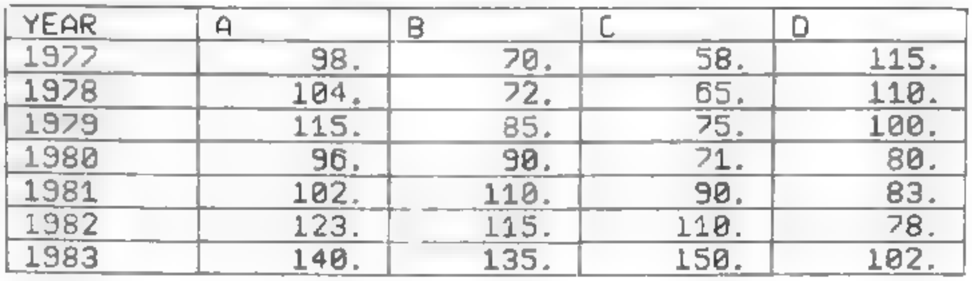
Hatching designation:	Quick, monochrome, color
Hatching color:	Blue, green, red (for color hatching)
Stack:	Maximum of 20 items (19 items if the first item is a character item)
Data range:	1E–98 ≦ l data l ≦ 1E98 or data = Ø

(2) Line Graph

The line graph is suited for showing changes or trends in time.

Use of the line graph is effective in the production, management, or sales division to show the changes in production or sales as trends.

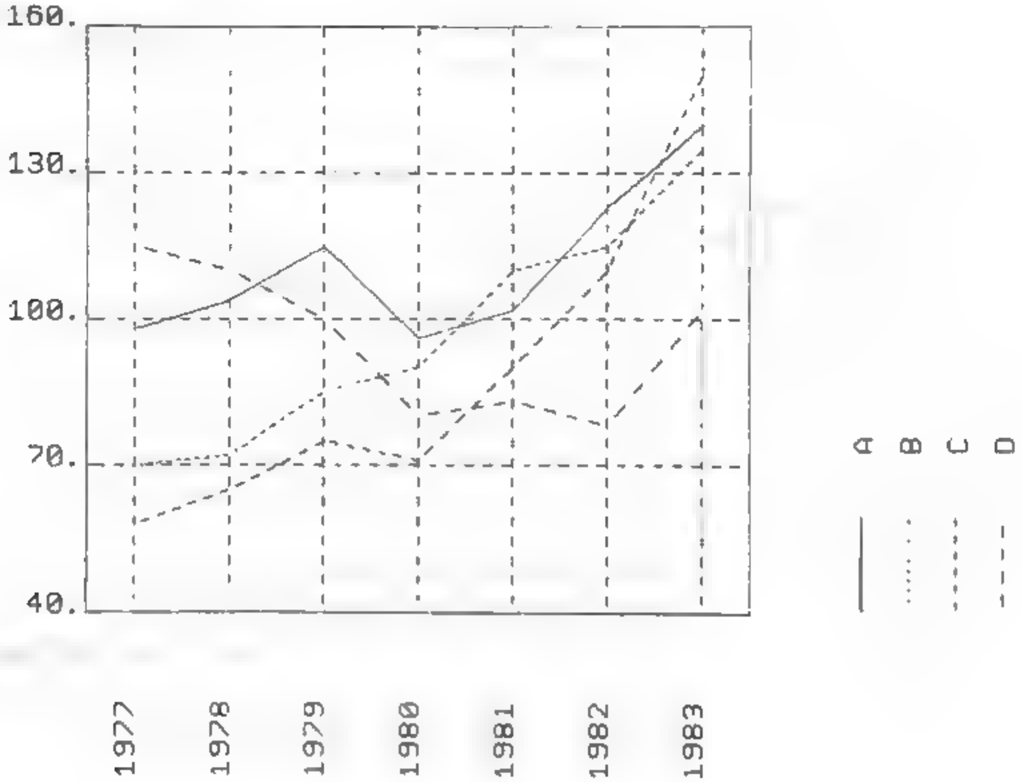
Example: Comparative sales of 4 companies



YEAR	A	B	C	D
1977	98.	70.	58.	115.
1978	104.	72.	65.	110.
1979	115.	85.	75.	100.
1980	96.	90.	71.	80.
1981	102.	110.	90.	83.
1982	123.	115.	110.	78.
1983	140.	135.	150.	102.

```
"YEAR":A:B:C:D
```

⟨ Print sample ⟩



(Print Conditions)

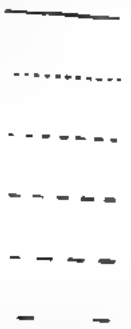
- Stack the sales of 4 companies
- Plot in black
- Draw scale lines
- MAX = A
- MIN = 40

= **Functions** =

Line designation: Black (quick, monochrome), color

Line color: Blue, green, red, black (when color is specified)

Type of lines:



Stack: Maximum of 20 items (19 items if the first item is a character item)

Data range: 1E−98 ≦ data ≦ 1E98 or data = 0

(3) Band Graph

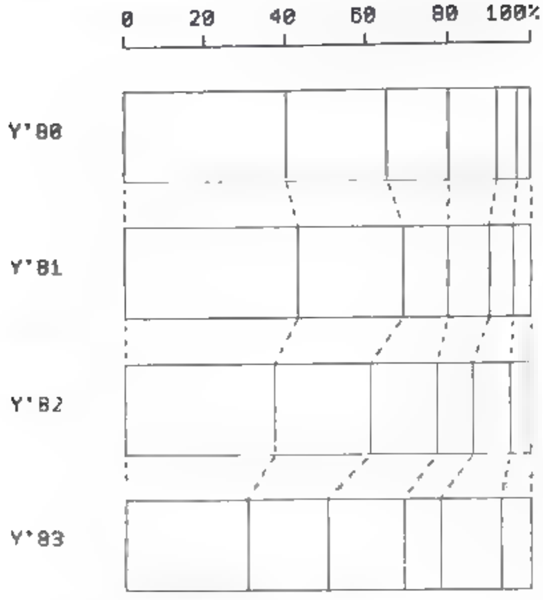
The band graph shows the relative composition. Since it's easy to see percentages in band form, changes in the relative composition can be shown by 2 or more band graphs.

The band graph can also be used to compare the compositions between different things.

(Example): Sales (in units) for each company.

Co.	Y'80	Y'81	Y'82	Y'83
A	31400.	45232.	48651.	48125.
B	19625.	27349.	31557.	32083.
C	11775.	11571.	21038.	30479.
D	9420.	10549.	11834.	14437.
E	3925.	6311.	11830.	24062.
F	2355.	4208.	6574.	11229.

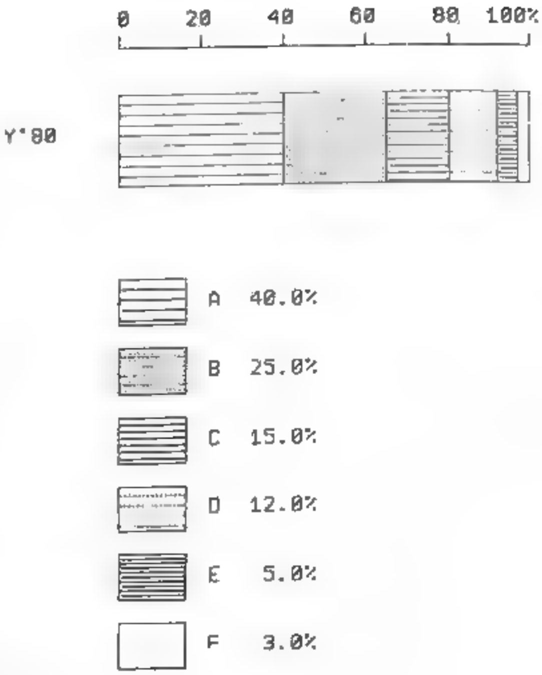
(Print Sample)



(Print Condition)

- Stack 4 items
- Quick

(Print Sample)



(Print Condition)

- No stacking
- Black hatching pattern for shading.

= **Functions** =

Hatching designation: Quick, monochrome, color

Hatching color: Blue, green, red, black (for color hatching)

Type of hatching:

Stack: Maximum of 20 items (19 items if the first item is a character item)

Data range: 1E−98 ≦ data ≦ 1E98 or data = Ø
(except when all data is Ø)

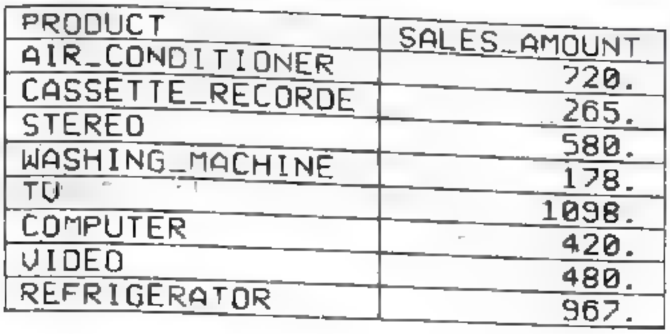
- Data total of each item > 3.6E−98

Percentage: Printed after the legends when printed individually (no stacking).

- The legends are not printed if the 0 key is pressed when specifying the hatching. The percentages are also not printed.

(4) Pie Graph

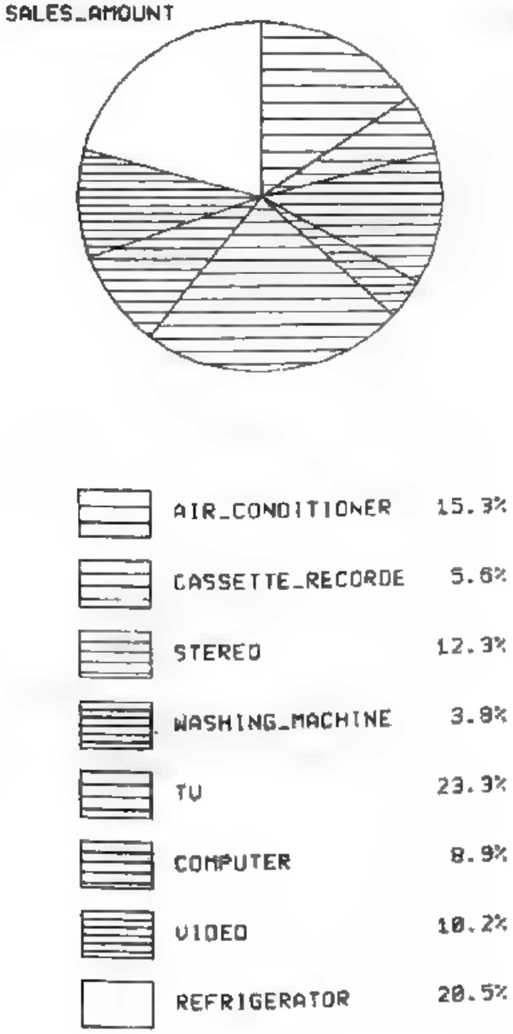
The pie graph (or pie chart) shows the individual components of a whole where the circle represents the whole. It's easy to see the percentages by comparing the area of the sectors. The compact form of the pie graph effectively presents percentages in a report such as for advertising.



PRODUCT	SALES_AMOUNT
AIR_CONDITIONER	720.
CASSETTE_RECORDE	265.
STEREO	580.
WASHING_MACHINE	178.
TV	1098.
COMPUTER	420.
VIDEO	480.
REFRIGERATOR	967.

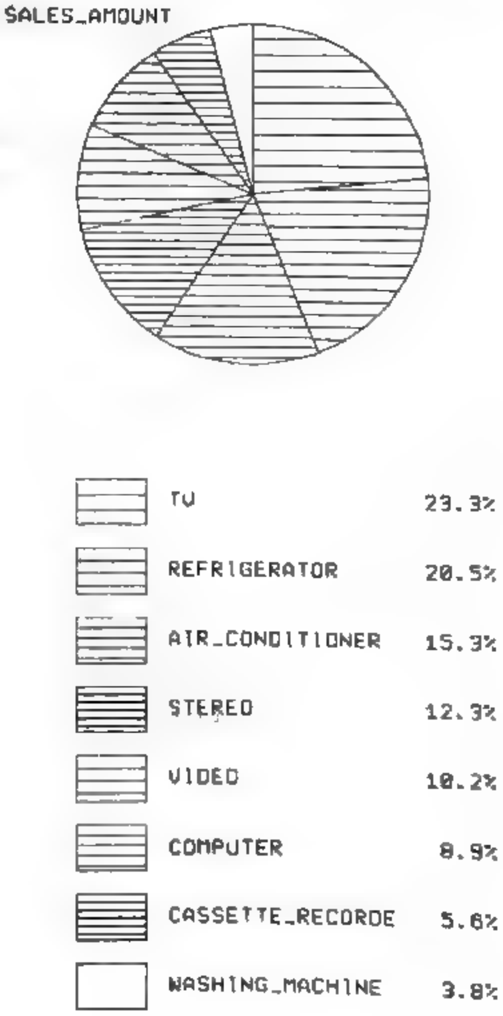
⟨FORMULA⟩ "PRODUCT":SALES_AMOUNT

⟨Print Sample⟩



⟨Print Sample⟩

Data sorted in order of sector area.



= **Functions** =

Hatching designation: Quick, monochrome, color
Hatching color: Blue, green, red, black (for color hatching)
Types of hatching: ▤ ▤ ▤ ▤ □
Stacking: None
Data range: 1E−98 ≦ data ≦ 1E98 or data = Ø
- Data total of each item > 3.6E−98

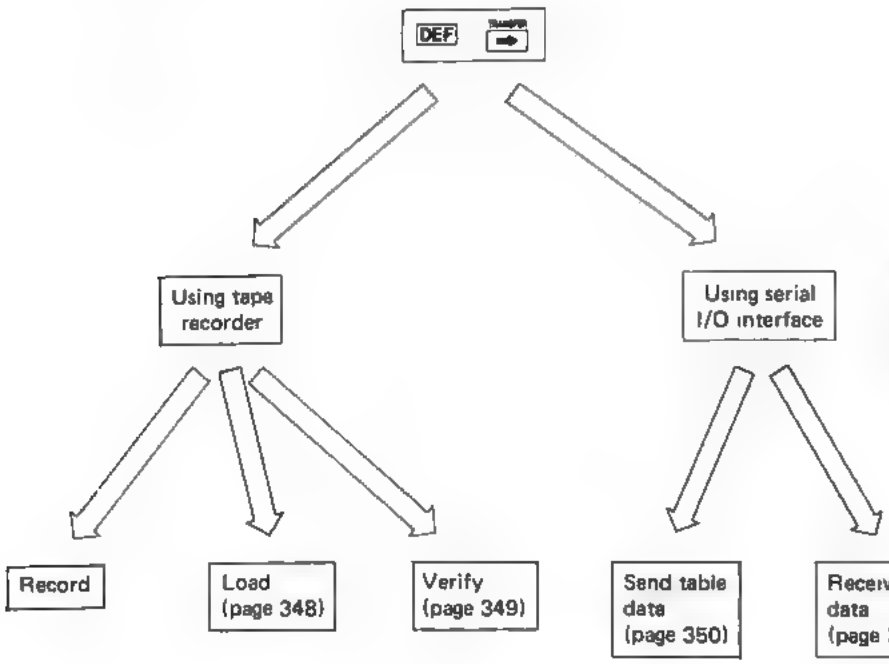
Percentage: Printed after the legends
- The legends are not printed if the [0] key is pressed when specifying the hatching. The percentages are also not printed.

3-8) Transferring a Table

The formula or table you created can be recorded (saved) on and later loaded from cassette tape.

Further, the formula or table can be sent or received through the serial I/O interface.

⟨ **Procedure** ⟩


DEF
TRANSFER
Using tape recorder
Using serial I/O interface
Record
Load (page 348)
Verify (page 349)
Send table data (page 350)
Receive table data (page 351)

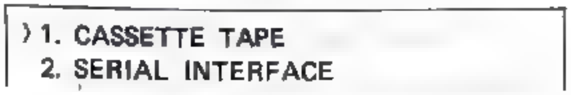
See the section on serial I/O interface under BASIC (page 83)

[1] Recording on Tape

A table or graph can be drawn after loading the table data from tape whenever necessary if you record it beforehand on tape. If you record it on tape, it's also easy to change any part of the data.

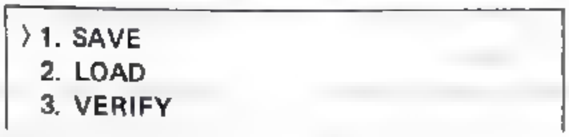
- Connect the tape recorder and advance the tape to the desired recording position (see page 83). When doing this, set the remote switch to OFF.
- Set the remote switch to ON, and press the record button on the tape recorder to prepare for recording. Follow the procedures below.

(1) DEF →



```
〉1. CASSETTE TAPE
  2. SERIAL INTERFACE
```

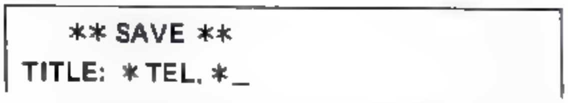
(2) 1 (or ENTER)



```
〉1. SAVE
  2. LOAD
  3. VERIFY
```

Press
1. The 1 key to save on tape
2. The 2 key to load from tape
3. The 3 key to ver fy

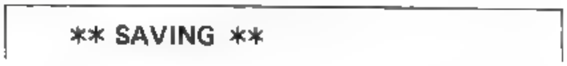
(3) Press the 1 key to save on tape.



```
    ** SAVE **
TITLE: *TEL.*_
```

Note: The title of the last table specified for table creation, graph creation, or table selection is displayed.

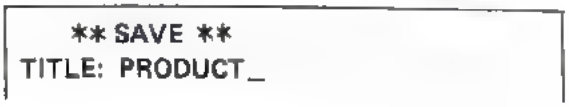
(4) If you want to save the table having the displayed title, press the ENTER key. The save operation begins.



```
    ** SAVING **
```

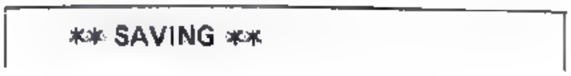
(5) If you want to save a different table, press the CLS key to clear the title. Enter the title of the table you want to save.

(Example) CLS "PRODUCT"



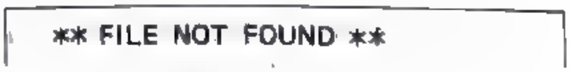
```
    ** SAVE **
TITLE: PRODUCT_
```

(6) Press the ENTER key. The save operation begins.



```
    ** SAVING **
```

Note: If the table for the title you input does not exist, the screen shown on the right will be displayed. The screen then returns to the one shown in step 3 above.



```
  ** FILE NOT FOUND **
```

- When the save operation ends, the screen returns to the one shown in step 2.
- After the save operation, verify the table. (See page 349.)

Jot the value of the tape counter when you begin saving so that you can easily find the start of the saved table data.

[2] Loading from Tape

Load the table data from the tape into the PC-2500.

- Advance the tape to the beginning of the saved table data. Turn the remote switch on and press the play button on the tape recorder to set it to the playback mode.
- Set the volume of the tape recorder to its maximum setting. (For tape recorders with a tone control, set the tone to its highest setting.)

(1) DEF →

```
) 1. CASSETTE TAPE
  2. SERIAL INTERFACE
```

(2) 1

```
) 1. SAVE
  2. LOAD
  3. VERIFY
```

(3) 2

```
      ** LOAD **
TITLE:
```

(4) Enter the title of table data to be loaded.

(Example) PRODUCT

```
      ** LOAD **
TITLE: PRODUCT
```

(5) ENTER (Starts the load operation)

```
      ** LOADING **
```

- The screen shown on the right will be displayed when the title of the table to be loaded is found. If a different title is found

  Skip:

  is displayed and the table is ignored.

```
      ** LOADING **
LOAD: PRODUCT
```

- If you press the ENTER key without entering the title in step 4, the tape starts to play and the table data of the first title found will be loaded.
- You cannot load two tables having the same name. If the specified title is the same as the title of a table already in the PC-2500,

  ** SAME FILE EXISTS **

will be displayed and the screen returns to step 3. Even if the title was not specified (only the [ENTER] key was pressed), the same message appears when the title of the table data to be loaded from tape is the same as the title of a table already in the PC-2500.

[3] Verifying

The verify operation compares the contents of the tape with the contents of the PC-2500 to confirm that the table was saved properly.

- Rewind the tape until the starting position of where the table was saved. Turn the remote switch to ON and press the play button on the tape recorder to set it to the playback mode.
- Set the volume of the tape recorder to its maximum setting. (For tape recorders with the tone control, set the tone to its highest setting.)

(1) [DEF] [→]

```
〉1. CASSETTE TAPE
  2. SERIAL INTERFACE
```

(2) [1]

```
〉1. SAVE
  2. LOAD
  3. VERIFY
```

(3) [3] (Select the verify operation)

Check whether the title is the one for the table to be verified. If it is not, clear it with the [CLS] key and enter the correct title.

```
          ** VERIFY **

TITLE: PRODUCT _
```

(4) [ENTER] (Start the verify operation)

```
          ** VERIFYING **
```

- When the title of the table to be verified is found, it is displayed as shown on the right. If a different title is found,
  Skip:
  is displayed and the table is ignored.

When the verify operation ends,

** END OF VERIFY **

will be displayed and the screen returns to the one shown in step 1 above.

If the data within the PC-2500 and the data saved on tape is found to be different during the verify operation,

** VERIFY ERROR **

will be displayed and the verify operation stops.

If the same verify error continues to recur after several attempts, save the table again.

- If the specified title is not found, the PC-2500 continues to search for it even after the end of the tape is reached. Press the [ON/BRK] key to stop. This also applies when loading table data.

[4] Transferring Table Data Through the Serial I/O Interface

Table data is transferred by using the serial I/O interface.

(1) Sending Table Data

(1) [DEF] [→]

```
〉1. CASSETTE TAPE
 2. SERIAL INTERFACE
```

(2) [2] (Selects the serial I/O interface)

```
〉1. SEND
 2. RECEIVE
```

Press the [1] key to send.
Press the [2] key to receive.

(3) [1] (Select send)

```
        ** SEND **
TITLE: AGE_
```

(4) Check the title. If you want to send a different table, press the [CLS] key and enter the correct title.

(5) [ENTER]
Starts the send operation.

```
        ** SENDING **
```

- If the connected equipment is not ready to receive data (e.g. power not turned on), no data is being sent even though

  ** SENDING **

  is displayed.
  Press the [ON/BRK] key and stop operation. Prepare the connected equipment and send again.

In order to transfer (send or receive) table data using the serial I/O interface, a connection device is required between the PC-2500 and the connected equipment.
Further, you must set the I/O conditions with BASIC's OPEN command (see page 202).
For information on the serial I/O interface, see page 233.

(2) Receiving Table Data

(1) [DEF] [→]

```
〉1. CASSETTE TAPE
 2. SERIAL INTERFACE
```

(2) [2]

```
〉1. SEND
 2. RECEIVE
```

(3) [2] (Select the receive operation)

```
        ** RECEIVE **
TITLE: *
```

(4) Enter the title of the table to be received.
If you do not specify any title, the first table data is loaded.

(5) [ENTER]
Starts the receive operation.

```
        ** RECEIVING **
```

- If the connected equipment is not ready, table data cannot be received and

  ** RECEIVE ERROR **

  will be displayed. The receive operation will stop.
  (Pressing the [ON/BRK] or [ENTER] key returns the screen to the one shown in step 3.)
  Prepare the connected equipment and start the receive operation again.

  Note: When

  ** RECEIVE ERROR **

  is displayed, be sure to clear the error with the CLS key before changing to another function. The print function may become inoperative if you press [DEF] and a business program key without clearing the error.

3-9) Special Functions

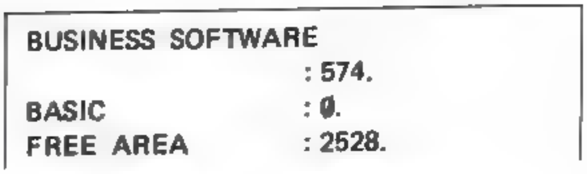
The following describes the methods for clearing a stored table, viewing the memory capacity and setting the use of the RAM card.

Press [DEF] [ON/BRK]. The screen on the right will be displayed.

```
〉1. DISPLAY FREE AREA
 2. DELETE FILE
 3. DELETE ALL FILES
 4. USE RAM CARD
```

[1] Checking the Memory Capacity

(1) Press the [1] key to display the memory capacity.



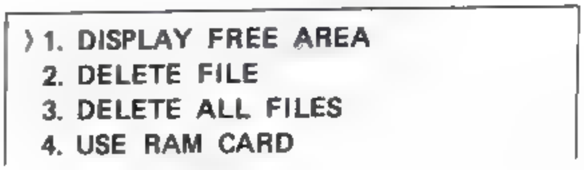
```
BUSINESS SOFTWARE
                     : 574.
BASIC                : 0.
FREE  AREA           : 2528.
```

The business program uses 574 bytes of memory.
BASIC does not use any memory.
The remaining capacity is 2528 bytes.
Note: The values change when the RAM card is used.

Memory usage is described on pages 172 and 354.

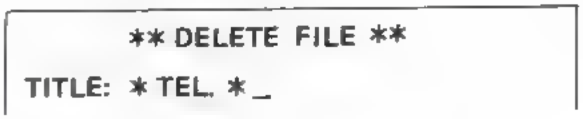
[2] Deleting a Table

(1) [DEF] [SYSTEM [illegible]]



```
〉1. DISPLAY FREE AREA
  2. DELETE FILE
  3. DELETE ALL FILES
  4. USE RAM CARD
```

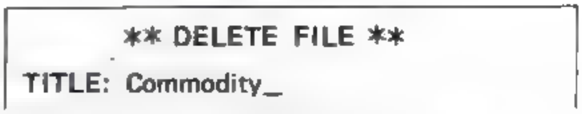
(2) [2]



```
        ** DELETE FILE **

TITLE: * TEL. *_
```

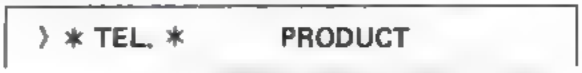
(3) Press the [ENTER] key to delete the table having the title displayed on the screen. To delete a different table, press the [CLS] key and enter the title of the table to be deleted.

(Example) [CLS] Commodity



```
        ** DELETE FILE **

TITLE: Commodity_
```

[ENTER] (Deletes the table)



```
 〉* TEL. *        PRODUCT
```

Note: If the table for the entered title does not exist

** FILE NOT FOUND **

is displayed and the screen returns to the one shown in step 2.

- If * Telephone List * was specified, only its data will be deleted and the title remains.

  After the data has been deleted, data for * Telephone List * can be loaded from tape.

[3] This function deletes all the table created using the business program.

(1) [DEF] [→]

```
)1. DISPLAY FREE AREA
 2. DELETE FILE
 3. DELETE ALL FILES
 4. USE RAM CARD
```

(2) [3]

```
 1. DELETE
)2. CANCEL
```

(3) Press the [1] key to delete.
Press the [2] key to cancel and return to the screen shown in step [1].

Note: The title * Telephone List * will not be deleted. (Its data will be deleted.)

If you delete all the tables and then enter BASIC, the program/data area that can be used by BASIC is at its maximum.

[4] Setting the RAM Card Functions

The RAM card has the following uses:

(a) To expand the memory capacity of the PC-2500. (Used as an expansion memory)

(b) To store table data as a data card. The RAM card can be substituted with another card.

This function is used to specify one of the above uses.

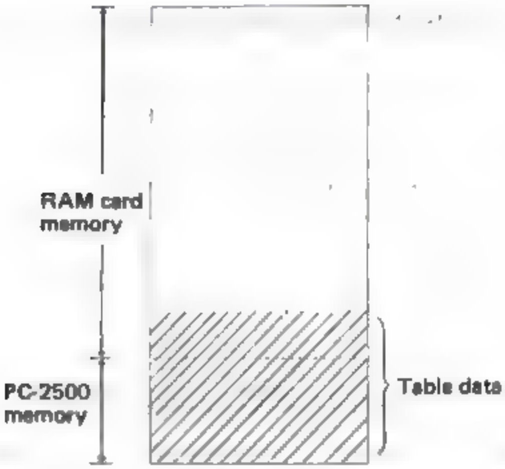
(1) [DEF] [illegible]

```
)1. DISPLAY FREE AREA
 2. DELETE FILE
 3. DELETE ALL FILES
 4. USE RAM CARD
```

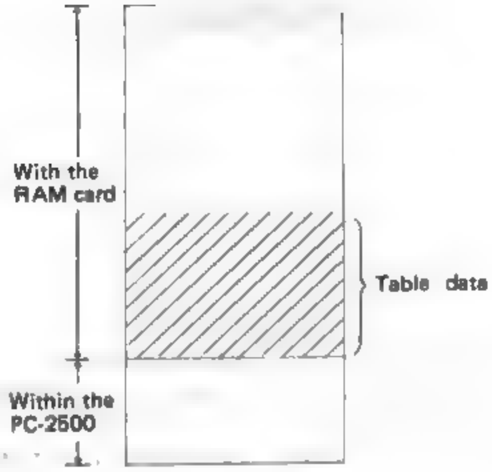
(2) [4]

```
)1. FOR EXPANSION
 2. FOR FILE CARD
```

(3) Press the [1] key to use the card as an expansion memory for the PC-2500.
Press the [2] key to use the card as a data card.



Use as an expansion memory: Table data is not retained when the RAM card is removed.



Use as a data card: Table data is retained within the RAM card even if it is removed. The RAM card stores the table data.

If the table data or BASIC program is too large and does not fit within the RAM card (when used as a data card),

```
** RAM CARD ERROR **
```

will be displayed.

- This function corresponds to the MEM "B" or MEM "C" commands in BASIC. See page 172.

Note: Even though there is memory remaining,

```
** MEMORY OVERFLOW **
```

may be displayed when you are checking the data, printing a table, or creating a graph from table data loaded from the tape recorder, or when you are creating a table. Various calculations and comparisons are performed within the PC-2500 to allow you to create a table or graph. The PC-2500 requires temporary memory area to store characters and numeric values. (This memory is called the working area.)

After this area is allocated, the above message appears when there is not enough memory space. You must clear some of the memory by deleting a table, or saving table data on tape and then deleting it.

If you are using both BASIC and the business program, you can also delete the BASIC program and its variables to clear more space.

4. ERROR AND WARNING MESSAGES

Many messages are displayed during use of the business program. This section deals with error and warning messages that are displayed when there is an error in the formula or data.

(1)

```
        ** SELECT FILE **
```

Table has not been specified.

- After this message, the telephone list will be automatically specified.

(2)

```
      ** FORMULA ERROR **
```

(1) There is an error in the formula during its input.

- The display to input the formula appears again.
  Correct the formula.

(2) There is an error in the equation for the formula and table calculation cannot be performed.

Example: a = $\underset{\uparrow}{\sim}$ b + c) 2

Missing "("

- Clear the table (using special function 2). Enter the correct formula and create a new table.

(3)

```
        ** INPUT ERROR **
```

This error occurs in the following cases:

○ Attempt was made to enter characters into a numeric item.

○ When the data is entered in equation form and the equation cannot be calculated.

Example: 5/0 is input.

○ Data greater than or equal to 1 x 10^{100} was input. Or inappropriate data was input.

Enter appropriate data.

(4)

```
   ** CALCULATION ERROR **
```

This error occurs in the following cases during table creation or data write:

○ The result of the equation in the formula is greater than or equal to 1 x 100^{100}

○ Illogical calculation such as division by 0.

Example: For a = b/c, when 0 was input for c or any illogical calculation or calculation resulting in a value beyond the calculable range.

*****

is entered for the item which is to store the result of the equation.

- If there is an error in the data, enter the correct data.
- If there is an error in the formula, delete the table, enter the correct formula, and create a new table.

(5)

```
** TOO MANY COLUMNS **
```

This message is displayed if there are too many items in the table when the formula is converted to table form.

- Change the formula so that there are 20 or less items.

(6)

```
** INVALID DATA **
```

This error occurs in the following cases during creation of a band graph or pie graph:

○ Negative value exists in the data.

○ Data values are all 0.

Band graph or pie graph cannot be created. Change the data.

○ No item is specified for the graph (all specified items cleared).

- Specify the items to be graphed.

(7)

```
** SAME TITLE EXISTS **
```

This message is displayed when an attempt is made to create a new table using a title for a table already in the PC-2500 or when a table having the same title is loaded from tape.

- During input of the formula change the first item.
  Or delete the table in the PC-2500.

(8)

```
** FILE NOT FOUND **
```

This is displayed in the following cases:

○ The table having the specified title does not exist in the PC-2500 when performing the save, verify, or send operation.

○ The table having the specified title does not exist when deleting a table using special function 2.

Check the title in the PC-2500 and enter the correct title. This error occurs when the table data saved on tape cannot be loaded properly.

(9)

```
** LOAD ERROR **
```

Clear the error with the CLS key and start the load operation again from the beginning.

If the error continues to recur:

1. The tape recorder head may be dirty.
2. Part of the table data saved on tape may have been erased.
3. Part of the tape is scratched, dirty, or wrinkled so that it cannot be loaded.
4. There may be large variations in tape speed.

Check by cleaning the head or using another tape.

(10)

```
** VERIFY ERROR **
```

This error occurs when the table data saved on tape does not match the data in the PC-2500 during the verify operation. Clear the error with the CLS key and verify again.

If the error continues to recur, save the table again and verify again.

As described for load error above, the cause of the error may lie in the tape recorder or tape.

(11)

```
** RECEIVE ERROR **
```

This error occurs when the table cannot be received properly through the serial I/O interface.
Clear the error with the CLS key and receive again.

(12)

```
** BREAK **
```

This message is displayed when the BREAK key is used to stop the save, load, verify, send, or receive operation during the transfer of a table.

- The display before execution of the operation is immediately restored.
- This message is also display when BREAK is pressed during printing tables or creating graphs.

(13)

```
** MEMORY OVERFLOW **
```

This message is displayed when there is insufficient memory within the PC-2500 and

○ The business program cannot be started
○ Working area for graph creation, table creation, etc. cannot be allocated
○ A new line for data input cannot be created.
○ A new table cannot be created, etc.

- Use special function 2 and delete the unnecessary table (or save them on tape and then delete).
  Or, delete the variables used in BASIC.

(14)

```
** RAM CARD ERROR **
```

This message is displayed when the PC-2500 cannot be set for use with the RAM card removed (during function setting for the RAM card).
(A capacity larger than that of the RAM card is being used. Use special function 2 and delete the unnecessary tables (or save them on tape and then delete). Or, delete the variables used in BASIC.

(15)

```
** ILLEGAL COLUMN **
```

This error occurs when an attempt is made to graph a character item.

(16)

```
* DRAW WITHOUT LEGEND *
```

This message is displayed when the key (0 key when specifying hatching) is pressed to specify that the graph is to be printed without legends. Usually, the graph is printed with legends.

(17)

```
** LOW BATTERY **
```

This message is displayed if the voltage of the battery is low when the printer is operated.
If this message appears, immediately charge the battery.

(18)

```
** TOO MANY DATA **
```

This message displayed when the number of data items for a column exceeds 50.

5. ACTION TO BE TAKEN FOR INCORRECT OPERATION

The following describes the actions to be taken in the event you make a mistake when pressing the keys while using the business program.

Situation	Action
(1) Messages such as ** Incorrect Formula ** ** Incorrect Input ** ** Impossible Calculation ** are displayed.	See the section "Error and Warning Messages."
(2) ● You inadvertently converted the formula into a table by pressing the ENTER key while in the middle of inputting the formula. ● You want to change the formula after having converted it to a table.	Once a formula has been converted to a table, it cannot be changed. Use special function 2. and delete the table. Re-input the formula and create the table.
(3) Data in the table is incorrect.	Call the table by pressing DEF WRITE ↑ . Move the index to the line containing the incorrect data and enter the correct data.
(4) You specified the wrong item and pressed a character or number key when entering data into the table.	If you have not pressed the ENTER key, press the [illegible] key to clear the data input mode. Move the index to the correct line and re-enter the data.
(5) You specified the wrong table and are in the mode to write data, specify the table format, or create and print a graph.	Press DEF ↓ to obtain the table selection screen and select the table. Re-enter the data write mode, etc.
(6) After the DEF key, you pressed the wrong key instead of a key for the business program.	Press the DEF key and then the business program key when the printer stops writing the table or graph. The pen moves to the left margin.

Situation	Action
(7) You are in the mode to print the graph or table and want to exit the mode. You want to stop printing.	Hold down the ON/BRK key for a while. Release the key when the printer stops writing the table or graph. The pen moves to the left margin.
(8) During graph creation, you selected the wrong graph.	Press DEF MODE and return to the graph selection screen. Select the correct graph.
(9) You made a mistake when specifying a graph item.	● Press the 1 key when "Stack?" is displayed and change to the item name selection screen. Move the index to the position of the incorrectly specified item and press the ENTER key. This clears the item. ● If the conditions are displayed or if you are now setting the conditions, press DEF MODE . Specify again from the beginning.
(10) You made a mistake when specifying the graph conditions.	Press DEF MODE and return to the graph selection screen. Specify again from the beginning.
(11) You selected the serial I/O interface by mistake for the table transfer. Or you selected the tape recorder by mistake.	When the selection screen "send/receive" or "system to cassette, cassette to system/verify" is displayed, press the ENTER key. The screen then requests input of the title. Press the ON/BRK key to return to the selection screen for the cassette or serial I/O interface. Select the correct one.
(12) You want to stop the transfer of the table.	Press the ON/BRK key while ** SAVING **, ** LOADING **, or **SENDING **, indicating execution, is displayed. This stops execution.
(13) You entered "table deletion" by mistake.	● When the screen requests input of the title, press the ON/BRK key. If you have already executed the table deletion, recovery is impossible. Either re-input the formula or load the table from tape.

Situation	Action
(14) You executed the deletion of all tables by mistake.	Recovery is impossible. Either re-input the formula or load the table from tape.
(15) You want to go on to the next operation after checking the remaining memory.	Press the DEF key and then the business program key. Or press SHIFT + [illegible] .

MODEL PC-2500 AND PERIPHERALS
LIMITED WARRANTY

Sharp Electronics Corporation warrants each of these products to the original purchaser to be free from defective materials and workmanship. Under this warranty the product will be repaired or replaced, at our option, without charge for parts or labor, with the exception of supplies, such as batteries, ribbons, inked rollers, etc., when returned to a SHARP FACTORY SERVICE CENTER listed in the instruction booklet supplied with your product.

This warranty does not apply to cassette tapes, software programs or appearance items nor to any product whose exterior has been damaged or defaced, nor to any product subjected to misuse, abnormal service or handling, nor to any product altered or repaired by other than a SHARP FACTORY SERVICE CENTER. This warranty does not apply to any product purchased outside the United States, its territories or possessions.

The period of the warranty shall be ninety (90) days on parts and labor from the date of the original purchase.

This warranty entitles the original purchaser to have the warrantied parts and labor rendered at no cost for the period of the warranty described above when the unit is carried or shipped prepaid to a SHARP FACTORY SERVICE CENTER together with proof of purchase.

THIS SHALL BE THE EXCLUSIVE WRITTEN WARRANTY OF THE ORIGINAL PURCHASER AND NEITHER THIS WARRANTY NOR ANY OTHER WARRANTY, EXPRESSED OR IMPLIED SHALL EXTEND BEYOND THE PERIOD OF TIME LISTED ABOVE. IN NO EVENT SHALL SHARP BE LIABLE FOR CONSEQUENTIAL ECONOMIC DAMAGE OR CONSEQUENTIAL DAMAGE TO PROPERTY. SOME STATES DO NOT ALLOW A LIMITATION ON HOW LONG AN IMPLIED WARRANTY LASTS OR AN EXCLUSION OF CONSEQUENTIAL DAMAGE, SO THE ABOVE LIMITATION AND EXCLUSION MAY NOT APPLY TO YOU. IN ADDITION, THIS WARRANTY GIVES SPECIFIC LEGAL RIGHTS AND YOU MAY HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

SERVICE CENTER ADDRESS

SHARP ELECTRONICS CORPORATION
Sharp Plaza
Mahwah, New Jersey 07430
(201) 529-8200

SHARP ELECTRONICS CORPORATION
6478 Interstate 85
Norcross, Georgia 30093
(404) 448-5230

SHARP ELECTRONICS CORPORATION
430 East Plainfield Road
Countryside, Illinois 60525
(312) 482-9292

SHARP ELECTRONICS CORPORATION
Sharp Plaza, 20600 South Alameda St.,
Carson, California 90810
(213) 637-9488

SHARP ELECTRONICS CORPORATION
1205 Executive Drive
East Richardson, Texas, 75081
(214) 234-1136

SHARP

SHARP ELECTRONICS CORPORATION

CORPORATE HEADQUARTERS AND EXECUTIVE OFFICES:
Sharp Plaza, Mahwah, New Jersey 07430. Phone:(201)529-8200
REGIONAL SALES OFFICES AND DISTRIBUTION CENTERS:
Eastern: Sharp Plaza, Mahwah, New Jersey 07430. Phone:(201)529-8200
Midwest: 430 East Plainfield Road, Countryside, Illinois 60525. Phone:(312)482-9292
Western: Sharp Plaza 20600 South Alameda Street, Carson, California 90810. Phone:(213)637-9488

SHARP CORPORATION
OSAKA, JAPAN

Printed in Japan
5M1.7T(TINSE4345CCZZ)③